Open access • Proceedings Article • DOI:10.1109/CEC.2016.7743797

Many-objective genetic programming for job-shop scheduling — Source link

Atiya Masood, Yi Mei, Gang Chen, Mengjie Zhang

Institutions: Victoria University of Wellington

Published on: 01 Jul 2016 - Congress on Evolutionary Computation

Topics: Job shop scheduling, Genetic programming and Tardiness

Many-Objective Genetic Programming for Job-Shop Scheduling

by

Atiya Masood

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Doctor of Philosophy
in Computer Science.

Victoria University of Wellington
2020

Abstract

The Job Shop Scheduling (JSS) problem is considered to be a challenging one due to practical requirements such as multiple objectives and the complexity of production flows. JSS has received great attention because of its broad applicability in real-world situations. One of the prominent solution approaches to handling JSS problems is to design effective dispatching rules. Dispatching rules are investigated broadly in both academic and industrial environments because they are easy to implement (by computers and shop floor operators) with a low computational cost. However, the manual development of dispatching rules is time-consuming and requires expert knowledge of the scheduling environment. The hyper-heuristic approach that uses genetic programming (GP) to solve JSS problems is known as GP-based hyper-heuristic (GP-HH). GP-HH is a very useful approach for discovering dispatching rules automatically.

Although it is technically simple to consider only a single objective optimization for JSS, it is now widely evidenced in the literature that JSS by nature presents several potentially conflicting objectives, including the maximal flowtime, mean flowtime, and mean tardiness. A few studies in the literature attempt to solve many-objective JSS with more than three objectives, but existing studies have some major limitations. First, many-objective JSS problems have been solved by multi-objective evolutionary algorithms (MOEAs). However, recent studies have suggested that the performance of conventional MOEAs is prone to the scalability challenge and degrades dramatically with many-objective optimization problems (MaOPs). Many-objective JSS using MOEAs inherit the same challenge as MaOPs. Thus, using MOEAs for many-objective JSS problems often fails

to select quality dispatching rules. Second, although the reference points method is one of the most prominent and efficient methods for diversity maintenance in many-objective problems, it uses a uniform distribution of reference points which is only appropriate for a regular Pareto-front. However, JSS problems often have irregular Pareto-front and uniformly distributed reference points do not match well with the irregular Pareto-front. It results in many useless points during evolution. These useless points can significantly affect the performance of the reference points-based algorithms. They cannot help to enhance the solution diversity of evolved Pareto-front in many-objective JSS problems. Third, Pareto Local Search (PLS) is a prominent and effective local search method for handling multi-objective JSS optimization problems but the literature does not discover any existing studies which use PLS in GP-HH.

To address these limitations, this thesis's overall goal is to develop GP-HH approaches to evolving effective rules to handle many conflicting objectives simultaneously in JSS problems.

To achieve the first goal, this thesis proposes the first many-objective GP-HH method for JSS problems to find the Pareto-fronts of nondominated dispatching rules. Decision-makers can utilize this GP-HH method for selecting appropriate rules based on their preference over multiple conflicting objectives. This study combines GP with the fitness evaluation scheme of a many-objective reference points-based approach. The experimental results show that the proposed algorithm significantly outperforms MOEAs such as NSGA-II and SPEA2.

To achieve the second goal, this thesis proposes two adaptive reference point approaches (model-free and model-driven). In both approaches, the reference points are generated according to the distribution of the evolved dispatching rules. The model-free reference point adaptation approach is inspired by Particle Swarm Optimization (PSO). The model-driven approach constructs the density model and estimates the density of solutions from each defined sub-location in a whole objective space. Furthermore,

the model-driven approach provides smoothness to the model by applying a Gaussian Process model and calculating the area under the mean function. The mean function area helps to find the required number of the reference points in each mean function. The experimental results demonstrate that both adaptive approaches are significantly better than several state-of-the-art MOEAs.

To achieve the third goal, the thesis proposes the first algorithm that combines GP as a global search with PLS as a local search in many-objective JSS. The proposed algorithm introduces an effective fitness-based selection strategy for selecting initial individuals for neighborhood exploration. It defines the GP's proper neighborhood structure and a new selection mechanism for selecting the effective dispatching rules during the local search. The experimental results on the JSS benchmark problem show that the newly proposed algorithm can significantly outperform its baseline algorithm (GP-NSGA-III).

Acknowledgments

I would like to express my sincere gratitude to the Victoria University of Wellington for providing the Victoria Doctoral Scholarship for me to continue my research studies in New Zealand.

I would like to express my sincere gratitude to my supervisors, Dr. Gang Chen, Dr. Yi Mei, Dr. Harith Al-Sahaf, and Professor Mengjie Zhang for their exemplary guidance, support, and encouragement throughout the progress of my Ph.D. study. They have provided invaluable feedback to improve my research.

I also take this opportunity to express my gratitude to Dr. John Park, Dr. Deepak Singh, Dr. Michael Ikechi Emmanuel, Philippa Becroft, Dr. TJ Boutorwick, and Karen Commons for their cordial support, valuable information, and guidance, which helped me in my research activities through various stages.

I would also like to thank my research colleagues, the Evolutionary Computation Research Group (ECRG), and the Evolutionary Computation for Scheduling and Combinatorial Optimisation group (ECCO) for their constructive feedback and comments in my study and discussions.

My sincere gratitude also goes to my beloved husband, Masood, for his constant support, encouragement, and understanding throughout this research work. I would like to thank my mother for her prayer, unconditional love and care. My sincere thanks as well to my uncles, Dr. Jaffery and Muhammad Kalim, for their constant support, prayer, and encouragement throughout my study.

List of Publications

1. **A. Masood**, Y. Mei, G. Chen, M. Zhang, "Many-objective genetic programming for job-shop scheduling", *Evolutionary Computation (CEC) 2016 IEEE Congress, pp. 209-216, 2016*.

2. **Masood, A.**, Mei, Y., Chen, G., Zhang, M.: A PSO-Based Reference Point Adaption Method for Genetic Programming Hyper-Heuristic in Many-Objective Job Shop Scheduling. *ACALCI. Lecture Notes in Computer Science, vol. 10142, pp. 326–338 (2017).*

3. **Masood, A.**, Chen, G., Mei, Y., Zhang, M.: Reference point adaption method for genetic programming hyper-heuristic in many-objective job shop scheduling. In: *European Conference on Evolutionary Computation in Combinatorial Optimization. pp. 116–131. Springer (2018).*

4. **Masood A.**, Chen G., Mei Y., Zhang M.: Adaptive Reference Point Generation for Many-Objective Optimization Using NSGA-III. In: *Advances in Artificial Intelligence. AI 2018. Lecture Notes in Computer Science, vol 11320. pp. 358-370. Springer(2018).*

5. **Masood A**., Chen G., Mei Y., Al-Sahaf H., Zhang M.: Genetic Programming with Pareto Local Search for Many-Objective Job Shop Scheduling. In: *Advances in Artificial Intelligence. AI 2019. Lecture Notes in Computer Science, vol 11919. pp. 536-548. Springer(2019).*

6. **A. Masood**, G. Chen,Y. Mei, H. Al-Sahaf, M. Zhang, "Fitness-based Selection Method for Genetic Programming with Pareto Local Search

in Many-Objective Job Shop Scheduling", *(Evolutionary Computation (CEC) 2020 IEEE Congress on, pp. 1-8, 2020)*.

7. **A. Masood**, G. Chen,Y. Mei, H. Al-Sahaf, M. Zhang, "Adaptive Reference Point Generation based on Gaussian Process Model for Many-Objective Optimization". Submitted to IEEE Transactions on Evolutionary Computation. (2020), 14pp. *(under review)*.

Contents

List of Tables

List of Figures

List of Abbreviations

ACO	Ant colony optimization
A-NSGA-III	Adaptive-NSGA-III
COVERT	Cost over time
DD	Due date
DRA	Density model-based reference point adaptation
EAs	Evolutionary algorithms
EC	Evolutionary computation
EDAs	Estimation of distribution algorithms
EMO	Evolutionary multi-objective optimization
FDD	Flow due date
GAs	Genetic algorithms
GaP	Gaussian process
GP	Genetic programming
GP-A-NSGAIII	GP-adaptive-Non-dominated sorting GA-III
GP-HH	Genetic programming based hyper-heuristic
GP-NSGA-III	GP-non-dominated sorting GA-III

HV	Hypervolume
IGD	Inverted generational distance
JSS	Job shop scheduling
MaOPs	Many-objective optimization problems
maxF	Maximal flowtime
maxWT	Maximal weighted tardiness
MARP	Model-based adaptive reference points
MBEAs	Model-based evolutionary algorithms
mF	Mean flowtime
MOEAs	Multi-objective evolutionary algorithms
MOPs	Multi-objective optimization problems
MRT	Ready time of the machine
mWT	Mean weighted tardiness
NMRT	Ready time of the next machine
NOIQ	Number of operations in the queue
NOINQ	Number of operations in the next queue
NOPT	Processing time of the next operation
NP	Non-deterministic polynomial-time
NSGA-II	Non-dominated sorting genetic algorithm-II
NSGA-III	Non-dominated sorting GA-III
PLS	Pareto local search
PSO	Particle swarm optimization
PT	Processing time of the operation

RVEA	Reference vector-guided evolutionary algorithm
SPEA2	Strength Pareto evolutionary algorithm
SPT	Shortest processing time
SI	Swarm intelligence
RVEA*	Adaptive version of RVEA
TWT	Total weight tardiness
W	Weight
WIQ	Work in the queue
WINQ	Work in the next queue
WKR	Work remaining

Chapter 1

Introduction

This chapter provides an introduction to this thesis and its motivations, goals, contributions, and organization. The problem statement is provided first, followed by a discussion of the main limitations of existing literature. The research goals and major contributions of this thesis are then discussed. This chapter also provides a brief discussion of how this thesis structured.

1.1 Problem statement

Job Shop Scheduling (JSS) [165] is a non-deterministic polynomial-time (NP) hard combinatorial optimization problem [53] in which various manufacturing jobs are assigned to machines at particular times while trying to minimize several objectives such as makespan, mean flowtime, mean tardiness. JSS problems have received significant attention from both academics and industry experts. From an industry perspective, JSS is considered to be a good model for many manufacturing scenarios because scheduling has direct impacts on the production efficiency and costs of a manufacturing system [94]. As reported by Johns and Rabelo [94], thousands of manufacturers contribute billions of dollars to the United State's economy. Furthermore, JSS is considered one of the significant production

scheduling problems in practice. It has a wide range of applications in many industries such as cloud computing [188] and management and operations research [115]. JSS has received substantial research attention due to its high computational challenges and strong practical value [143, 165].

A JSS problem usually has a set of *machines* on the *shop floor* that can be used to process a set of *jobs* [165]. Each job has a predetermined sequence of *operations*, which needs to be carried out in order to complete the job. Each job has a predetermined route through the machines before it leaves the shop. The machine can only process one operation at a time. JSS aims to process all arriving jobs by the machines in an optimal way so that the predefined objectives (e.g., makespan, and tardiness) are optimized in order to maximize the total revenue subject to the constraints (e.g., order of the operations of the jobs and the available machines for the jobs). The quality of a schedule in a job shop depends on the objective(s) of the problem.

JSS problems can be classified into two categories: *static* and *dynamic* [165]. The first subset belongs to static JSS problems where a problem has a fixed amount of jobs with known processing requirements such as processing time [165]. The second subset is dynamic JSS problems. In dynamic JSS problems, jobs arrive on the shop floor at various instances of time with no prior process information, e.g., release date, due date, and processing time [165].

Approaches for solving static JSS problems can be broken down into two main categories. The first category is the exact mathematical optimisation approaches [2, 19, 25, 55, 165]. The exact mathematical optimization approaches generate optimal solutions for JSS problem instances. The exact algorithms aim at searching through the full solution space and only work well on small-scale JSS problems [13]. For larger JSS problems, these algorithms are infeasible due to the exponential size of the solution space [19]. It is reported in [13] that the JSS problem with up to 100 jobs and 20 machines is computationally intractable for any exact mathematical opti-

mization approach. For such large-scale problems, finding near-optimal solutions (not optimal) within a reasonable amount of computation time is more feasible than finding optimal solutions with high computation cost. In this context, the second category of heuristic approaches is considered more desirable. These approaches are fast, although they cannot guarantee the optimality of the solution for JSS problems. These heuristic approaches are "rules-of-thumb" with the hope of generating "good" schedules [71]. Heuristic algorithms are mostly divided into two groups: the first group consists of meta-heuristics which directly targets the solution space (schedules). The second group explores the space of scheduling heuristics, typically in the form of dispatching rules represented as priority functions.

Meta-heuristic algorithms are search-based methods (e.g., genetic algorithms, particle swarm optimization). They can be used to search the solution space [9, 152, 157]. These methods are higher-level heuristics which provide a general framework to guide low-level heuristics that make local decisions. Meta-heuristic algorithms have the advantage that they can find high-quality solutions and successfully tackle large scheduling problems [152, 157]. A downside of meta-heuristic algorithms is that they require a substantial amount of time to explore high-quality solutions but are faster than exact algorithms, particularly for large problems [152].

Dispatching rules have been applied extensively to JSS problems due to their computational efficiency [176, 185]. Dispatching rules can be seen as a priority function which is used to assign priority to each job waiting to be processed by a machine. Then, the next job to process will be selected based on the priority value. Such computation is carried out at each decision point (e.g., when a machine becomes idle) and can be done efficiently [185]. There are two broad types of dispatching rules: (1) non-delay and (2) active rules. As the name indicates, non-delay rules do not allow any delay on the idle machines as long as the waiting queue is not empty [165]. On the other hand, active rules [165] allow some reasonable

delay (which is no more than the minimal processing time of the waiting jobs) to handle the potential new job arrivals with an urgent due date. Nguyen et al. [141] explored three different genetic programming (GP) representations of dispatching rules and proved that evolved dispatching rules could outperform benchmark human-made dispatching rules for JSS problems. Hunt et al. [75] evolved effective dispatching rules for the two-machine JSS problem to minimize the makespan. Dispatching rules are attractive to both researchers and practitioners because of their simplicity and high scalability in comparison to most of the direct optimization methods [141, 159]. However, in designing a dispatching rule for JSS, there are two main challenges. First, dispatching rules are time-consuming to design manually, especially for optimizing multiple potentially conflicting objectives, a frequent demand in a manufacturing environment. Meanwhile, any dispatching rules to be designed by domain experts will have to go through a lengthy and costly process [70]. Second, it is not always clear which of the existing dispatching rules is suitable for the given JSS problems [90].

In order to deal with these disadvantages, hyper-heuristics have been adopted in this thesis to design the dispatching rules automatically [70, 103]. GP has been a promising approach for designing dispatching rule heuristics automatically because GP has an ability to evolve priority functions with its flexible representation [143, 103]. The hyper-heuristic approach that uses GP to solve JSS problems is known as GP based hyper-heuristic (GP-HH) [17, 143].

A GP-HH evolves dispatching rules for both static and dynamic JSS problems automatically [17, 142, 143, 147, 145]. A GP-HH bypasses the need for human experts and extensive trial-and-error to construct dispatching rules. Dispatching rules evolved by GP can be directly and intuitively represented as tree-based priority functions. Moreover, there are a wide variety of evolutionary algorithms that deal with multi-objective (two or three objectives) and many-objective optimization (four or more

objectives), which shows to have good performances on many optimization problems. Therefore, by coupling GP with such multi-objective and many-objective algorithms can design a group of non-dominated dispatching rules automatically. These dispatching rules are expected to achieve a wide range of trade-offs over many conflicting objectives.

Since the early 2000s, the research on *multi-objective* JSS has started gaining popularity. Therefore, many independent studies have been carried out for multi-objective JSS with two ore three conflicting objectives, including the makespan, mean flowtime, maximum tardiness, maximum lateness, total workload, and proportion of tardy jobs [147, 184, 201]. In the literature, multi-objective optimization problems in a job shop were generally treated in two ways. The first approach aggregates multiple objectives together into a single objective through a weighted sum (for example, the linear weighted summation function). It then applies the single-objective optimization method to find the (single) optimal solution [79]. The second category of approach aims at finding a set of optimal solutions based on the Pareto-dominance concept [42] instead of finding a single optimal solution using aggregation functions.

The Pareto-dominance concept defines that a solution x dominates another solution y if no objective of x is worse than the corresponding one of y and at least one objective of x is better than y. Based on this concept, Pareto-optimal rules in JSS refer to the rules that cannot be dominated by others based on their performance on concerned scheduling objectives [116]. Multi-objective scheduling problems with the goal to find the Pareto-optimal front was first considered by Ishibuchi et al. [79] in 1998. After that, many evolutionary computation (EC) algorithms have also been proposed to evolve the Pareto-front for multi-objective JSS [44, 211].

Recently, many-objective optimization has become an active research topic [43, 116]. As emphasized by Deb in [43], a large proportion of real-world problems can be described naturally as many-objective optimization problems (MaOPs). Many-objective optimization refers to a class

of optimization problems that have more than three objectives. The last decade has witnessed the emergence of many-objective optimization as a booming topic in a wide range of complex modern real-world scenarios [84]. There is also a growing interest in industries to tackle problems with many objectives [43, 86]. According to our knowledge, the research on many-objective JSS algorithms is limited in the literature. Only a few algorithms in the literature tackles JSS problems consisting of more than three scheduling objectives [51, 147]. These algorithms utilize the conventional multi-objective evolutionary algorithms (MOEAs) (NSGA-II and SPEA2) but MOEAs experienced substantial difficulties when they were adopted to tackle MaOPs [43, 84]. These conventional MOEAs lost their selection pressure in solving MaOPs which is essential for the population to converge toward the Pareto-front [84]. Because of the lack of research on many-objective JSS, it is both theoretically and practically important to develop innovative GP-HH algorithms for many-objective JSS.

Hence, the overall goal of this thesis is to develop *GP-HH approaches to evolve effective dispatching rules for many conflicting objectives in JSS problems.*

1.2 Motivations

Scheduling theory has been established over the years, but most of the existing literature on the automatic design of dispatching rules mainly concentrates on single-objective JSS [67, 159] and multi-objective JSS [79, 200]. However, in practice, a good schedule for a JSS problem is expected to satisfy many objectives such as minimizing: tardiness, makespan, flowtime, and percentage of tardy jobs.

Most studies in JSS considered decomposition-based approaches to optimize these objectives together [67, 185]. In decomposition-based approaches, objectives are aggregated into a single objective (fitness function) and the problem is treated as a single-objective optimization problem. Even in the case where normalized objective functions are used,

the assigned weights still need to be predefined by the decision-makers and decision-makers should have good information about the trade-offs among different objective functions. To alleviate the strong dependency on domain knowledge, evolving the whole Pareto-front of non-dominated dispatching rules is naturally a better target for JSS problems with two or more than two objectives. It will also assist decision-makers in deciding of selection for quality dispatching rules which based on the trade-offs represented by the obtained Pareto-front.

Very few studies [129, 147] attempted to solve the many-objective JSS using MOEAs such as non-dominated sorting genetic algorithm II (NSGA-II) [44] and the strength Pareto evolutionary algorithm 2 (SPEA2) [211]. However, recent studies have suggested that conventional MOEAs are prone to the scalability challenge, i.e., the performance of these MOEAs degrades dramatically with the MaOPs [84, 43]. This is because MOEAs cannot provide sufficient selection pressure towards the Pareto-frontfor MaOPs. The main reason is that the number of non-dominated solutions increases exponentially as the number of objectives increases [84]. Many-objective JSS using MOEAs faces the same scalability challenges. Thus, using MOEAs for many-objective JSS often fail to select quality dispatching rules. Although the effectiveness of GP-HH has been extensively studied in the field of JSS [143, 147] (single JSS and multi-objective JSS), it was seldom explored in the context of many-objective JSS.

In MOEAs, when the primary selection criterion (convergence) cannot readily find good solutions, the diversity-based secondary criterion plays a vital role in the selection of the fittest solutions. For good solution diversity, MOEAs requires uniformly distributed candidate solutions to approximate the Pareto-front jointly [31]. However, in the case of a high dimension in terms of a number of objectives, it is quite challenging to maintain solution diversity because of the sparse distribution of the candidate solutions. The diversity maintenance can be controlled by providing multiple predefined reference points [43]. The reference points method

is one of the prominent and efficient ways for diversity maintenance in many-objective problems. With widely distributed reference points, solutions associated with these points are also encouraged to be distributed widely in the objective space.

Several recently developed algorithms such as NSGA-III [43], RVEA [31], and SPEAR [92] employed a predefined set of uniformly distributed reference points located on a hyperplane. These algorithms have successfully solved various practical MaOPs. One of the reasons for using a uniform distribution of reference points is because candidate solutions corresponding to each reference point can be emphasized to find a set of widely distributed sets of Pareto-optimal solutions. It is also reported by Ishibuchi et al. [80] that good results can be obtained by MOEAs algorithms only if the distribution of reference points is consistent with the Pareto-front shape of the problem to be solved. In this case, uniformly distributed reference points are inappropriate when the true Pareto-front is non-uniform, discontinuous, and irregular [86].

JSS problems often have irregular Pareto-front [142]. Therefore, many uniformly distributed reference points may be located in a region where no Pareto-optimal solutions exist and they are never associated with any dispatching rules on the evolved Pareto-front and become useless reference points. Clearly, if only a few reference points are truly associated with the dispatching rules evolved by GP-HH at the current generation, it will not be easy to distinguish and select these rules to improve diversity for future generations. In other words, many reference points will be associated with a large number of candidate dispatching rules. Some rules can be far from the corresponding reference point. Such a dispatching rule should enjoy higher selection opportunities but may not be selected during evolution simply because it is associated with a popular reference point. This issue has been clearly highlighted in [85, 86]. In particular, to evolve high-quality dispatching rules, it is essential to match the distribution of the reference points with the distribution of the Pareto-front. This

is one of the key research issues for many-objective JSS problems.

Pareto Local Search (PLS) is a simple and effective local search method for tackling multi-objective combinatorial optimization problems [26]. It is an extension of iterative local search algorithms for single-objective problems [149] to the multi-objective domain. PLS is a crucial component of memetic algorithms. Researchers have studied the application of PLS to multi-objective evolutionary algorithms with some success [26]. In fact, by hybridizing global search with local search, the performance of many MOEAs can be noticeably improved [26, 79]. Despite the preliminary success, the practical use of PLS on many-objective JSS is relatively limited. Moreover, our comprehensive search of the literature does not discover any existing studies involving the use of PLS in GP-HH for many-objective JSS.

In this thesis, one of our goals is to enhance the quality of evolved dispatching rules for many-objective JSS through hybridizing GP with the PLS technique. There are three challenges herein. First, the neighborhood structure in GP is not defined. Second, the PLS algorithm may suffer from low efficiency in the case of more than two objectives [89] because the number of optimal solutions grows exponentially with a number of objectives. Thus to search the neighborhood of a massive number of solutions becomes very time-consuming. For this reason, the size of the initial solutions for neighborhood exploration should be determined carefully. Third, the hybridized algorithm for many-objective JSS is how to divide the available computation time between the local search and the global search. A thorough investigation is required to determine how to achieve a desirable balance between local search and global search.

1.3 Research goals

The overall goal of this thesis is to develop GP-HH methods to evolve reusable and effective dispatching rules for many-objective JSS problems.

This research is also focused on investigating the conflicting nature among many related objectives that often should be considered together for effective JSS. This research aims to develop GP-HH methods that alleviate issues related to many-objective optimization in JSS problems and evolve new effective dispatching rules that are capable of enhancing the productivity of job shops. It is expected that the evolved rules can be reusable in unseen situations and outperform state-of-the-art multi-objective and many-objective algorithms in the literature. This research will be broken down into the following key objectives:

1. *Investigate how GP can be used to handle many-objective JSS problems.*

   This research objective explores whether the standard multi-objective optimization algorithms, such as NSGA-II and SPEA2, are competent at tackling many-objective JSS problems. To carry out this research effectively, we adopt a popular reference points-based approach (NSGA-III) [43] for many-objective optimization JSS . In particular, this research objective combines GP with the many-objective fitness evaluation scheme of NSGA-III and compares the performance of the resulting hybridized algorithm with NSGA-II and SPEA2 in solving many-objective JSS. It is expected that the newly proposed method will evolve high-quality Pareto-optimal dispatching rules that can effectively tackle not only the training problem instances but also unseen problem instances. The new algorithm is also investigating factors that influence the trade-offs among different objectives. In this objective, we will further study whether many typically considered scheduling objectives such as mean flowtime, mean total tardiness, maximum tardiness, and maximum flowtime are mutually conflicting or not.

2. *Investigate how to develop GP-HH approaches for the non-uniform Pareto-front of many-objective JSS problems which can evolve high-quality Pareto-optimal dispatching rules.*

This study is mainly focused on addressing the diversity issue in many-objective JSS. In the literature, diversity maintenance is often controlled by providing multiple predefined reference points [31, 43]. These points are widely distributed. Therefore, solutions associated with these points are also widely distributed in the objective space.

Most of the algorithms, such as NSGA-III and RVEA [31, 43] employ a predefined set of uniformly distributed reference points. However, as discussed in Section 1.1, the adoption of uniformly distributed reference points affect the solution diversity and deteriorate algorithm's performance. To address this vital issue of useless reference points, the main goal of this research objective is to develop new and effective mechanisms for reference point generation. These mechanisms improve the association between reference points and the Pareto-front during the whole evolutionary search process. Further, these new algorithms will be developed to increase the chance of discovering well-distributed solutions on the Pareto-optimal fronts. In this research objective, we explore both model-free and model-driven techniques to approximate the Pareto-front based on evolved dispatching rules accurately. In order to evaluate the performance of the proposed model-free and model-driven methods, we evaluate JSS problems and ten mathematical optimization benchmark problems. These problems have irregular, disconnected, degenerate, and inverted shapes of the Pareto-front where the number of objectives is scaled from three to eight. The proposed algorithms will be compared with four state-of-the-art algorithms as well as adaptive reference point algorithms.

3. *Investigate how to hybridize a local search with a global search and improve the quality of the evolved rules in many-objective JSS.*

   Many existing research studies analyzed the potential combination

of the local search with evolutionary multi-objective optimization (EMO) algorithms [26, 79, 108]. PLS has three phases: (1) the selection of initial solutions phase from the non-dominated solution's archive; (2) the exploration phase, in which the neighborhood of the selected solutions is explored, and candidate neighbors are extracted; and (3) the Pareto-archive phase, in which the archive is updated with the candidate neighboring solutions. The main goal of this study is to build an effective many-objective PLS algorithm for JSS problems.

Driven by this goal, this study thoroughly investigates the inclusion of PLS within GP-HH algorithms. Firstly, this research objective builds an effective strategy for selecting initial individuals. In particular, a fitness-based selection strategy will be proposed to balance convergence and diversity measures properly. By using this fitness-based selection strategy, we can select the representatives of each group of solutions. Secondly, the neighborhood structure of a tree (dispatching rule) in GP is defined. Specifically, this can be done by using restricted subtree mutation. This mutation can prevent a new neighboring rule discovered during the local search process from being significantly different from the original rule. Lastly, for comparing neighbor rule with its immediate parent, we consider the following two different strategies: (1) the scalarization strategy [79] and (2) the replacement strategy [26]. In the scalarization strategy, the objective vector of each rule is aggregated into a scalar using the weighted sum. The replacement strategy is based on the dominance relation.

It is evidenced in the literature [79] that the total number of generations for global search and the maximum number of local search steps is highly influential on the performance of hybridized algorithms. To further enhance the effectiveness of our new PLS algorithms, we investigate how to keep the balance between the number

of generations (for global search) and local search steps in our proposed algorithms. It is expected that our proposed algorithms help to enhance the exploitation ability and increase the chance of discovering promising dispatching rules.

1.4 Major contributions

This thesis makes the following major contributions.

1. This study proposes the first many-objective GP-HH method for JSS problems to find the Pareto-front of non-dominated dispatching rules by many conflicting objectives. The experimental results show that the combination of GP and NSGA-III produces a competitive algorithm as compared with NSGA-II and SPEA2 for evolving a set of trade-offs rules in many-objective JSS. The detailed analysis of the evolved Pareto-front reveals in-depth knowledge about the useful terminals for a large proportion of evolved dispatching rules. The rules evolved by the new algorithm also exhibit good generalization abilities as verified by performance on testing benchmark instances.

   Part of this contribution has been published in:

   A. Masood, Y. Mei, G. Chen, and M. Zhang, "Many-objective genetic programming for job-shop scheduling", *Evolutionary Computation (CEC) 2016 IEEE Congress on, pp. 209-216, 2016*.

2. This thesis presents two new approaches to dealing with the challenges caused by the non-uniform distribution of Pareto-front in objective space and proposes new adaptation mechanisms which improve the performance of evolutionary search and promote population diversity for better exploration. This contribution has the following sub-contributions:

(a) A new adaptive reference point strategy is proposed by using the adaptive model-free approach and the reference points are generated according to the distribution of the solutions. A new reference point adaptation mechanism has been successfully developed based on Particle Swarm Optimization (PSO). Essential changes to particle dynamics in PSO have also been introduced in our algorithm to prevent the majority of reference points from converging to small areas in the objective space. This reference point adaptation mechanism helps to reduce useless reference points and significantly improve the performance of the proposed algorithms. In the experimental evaluations based on the Taillard benchmark set, we successfully showed that the proposed reference point adaptation mechanism could significantly improve the performance of GP-HH and NSGA-III in terms of both HV and IGD.

Part of this contribution has been published in:

Masood, A., Mei, Y., Chen, G., and Zhang, M.: A PSO-Based Reference Point Adaption Method for Genetic Programming Hyper-Heuristic in Many-Objective Job Shop Scheduling. *ACALCI. Lecture Notes in Computer Science, vol. 10142, pp. 326–338 (2017).*

(b) This thesis develops a new adaptive strategy for generating reference points based on a model-driven technique. This model-driven technique estimates the density of solutions from each pre-defined sub-location in the entire objective space. Furthermore, the proposed algorithm provides smoothness to the model by applying a Gaussian Process on the density-based model which gives the ability to reduce the noise of the model and then calculate the area under the mean function. This area under the curve finds the required number of reference points in each sub-location. The experimental results demonstrate that

a new adaptive mechanism can outperform several state-of-the-art MOEAs such as RVEA and NSGA-III as well as cutting-edge MOEAs that support adaptive reference points.

Part of this contribution has been published in:

Masood, A., Chen, G., Mei, Y., and Zhang, M.: Reference point adaption method for genetic programming hyper-heuristic in many-objective job shop scheduling. In: *European Conference on Evolutionary Computation in Combinatorial Optimization. pp. 116–131. Springer (2018).*

Masood A., Chen G., Mei Y., and Zhang M.: Adaptive Reference Point Generation for Many-Objective Optimization Using NSGA-III. In: *Advances in Artificial Intelligence. AI 2018. Lecture Notes in Computer Science, vol 11320. pp. 358-370. Springer(2018).*

A. Masood, G. Chen,Y. Mei, H. Al-Sahaf, and M. Zhang, "A Model-Based Approach for Many-Objective Optimization Algorithm with Adaptive Reference Point Generation", *To be submitted for review to Genetic and Evolutionary Computation Conference (GECCO 2020).*

A. Masood, G. Chen,Y. Mei, H. Al-Sahaf, and M. Zhang, Adaptive Reference Point Generation based on Gaussian Process Model for ManyObjective Optimization". Submitted to IEEE Transactions on Evolutionary Computation. (2020), 14pp, *(under review.*

3. This research objective investigates the usefulness of the PLS in GP-HH and develops two new algorithms that combine GP-HH as a global search with PLS as a local search. This is the first algorithm of its kind for many-objective JSS that combines GP with PLS-based local search. In this objective, we propose the new selection mechanism for the initial solutions for neighborhood exploration in PLS. This mechanism defines representatives from each group of solu-

tions and selects a solution based on their fitness value. Further, the proposed algorithm defines the proper neighborhood structure for tree-based dispatching rules. A new selection mechanism for selecting the suitable dispatching rules during the neighborhood exploration is also defined. A detailed sensitivity analysis is also performed. The analysis finds the total number of generations and the maximum number of local search steps in GP-PLS.

Part of this contribution has been published in:

Masood A., Chen G., Mei Y.,H. Al-Sahaf, and Zhang M.: Genetic Programming with Pareto Local Search for Many-Objective Job Shop Scheduling. In: *Advances in Artificial Intelligence. AI 2019. Lecture Notes in Computer Science.*

A. Masood, G. Chen,Y. Mei, H. Al-Sahaf, and M. Zhang, "Fitness-based Selection Method for Genetic Programming with Pareto Local Search", *Evolutionary Computation (CEC) 2020 IEEE Congress on, pp. 1-8, 2020.*

1.5 Organisation of thesis

The remainder of this thesis is organized as follows. Chapter 2 presents a literature review of related works. Chapter 3 presents the experimental methodology of this thesis. Chapters 4–6, present the main contributions of the thesis. Chapter 7 concludes the thesis.

- *Chapter 2* presents descriptions of the JSS problem, many-objective optimization, local search, Pareto local search, GP, and dispatching rules used for this thesis. The basic concepts of meta-heuristics, heuristics, and hyper-heuristics for the automatic generation of heuristics are also presented. This chapter reviews current research methodologies using GP for the automatic generation of new

dispatching rules in single-objective, multi-objective, and many-objective JSS problems.

- *Chapter 3* presents the core experimental methodology of this thesis. It starts with the benchmark problems used throughout the thesis, the terminals and function set used to evolved dispatching rules. At the end of the performance measures for experimental results analysis are discussed in detail.

- *Chapter 4* develops a new many-objective GP-HH method to deal with the four conflicting objectives of JSS simultaneously. The proposed GP-HH method is used to evolve the non-dominated dispatching rules that are represented by the obtained Pareto-front. An extensive comparison between the evolved rules from GP-NSGA-III and the two famous MOEAs (GP-NSGA-II and GP-SPEA2) have been carried out. Popular multi-objective performance measures are also used to help assess the performance of the compared algorithms.

- *Chapter 5* identifies a key research issue involved in using the uniformly distributed reference points, i.e., the failure to promote solution diversity during evolution which affected the performance of GP-HH. To solve these issues, two new reference point adaptation mechanisms are proposed which improve the association between reference points and the evolved Pareto-front, enhancing solution diversity, and hence the performance of the algorithm. This chapter consists of two sections: (i) the first section develops an adaptive model-free approach and (ii) the second section proposes a model-driven approach for learning irregular distributions of the Pareto-front and systematically sampling reference points. The proposed algorithm introduces the model-based technique based on the Gaussian process model which estimates the density of solutions from each defined sub-location in a whole objective space. This density-

based model learns the distribution of the candidate solutions and accurately approximates the Pareto-front based on the evolved solutions. To verify the performance of the proposed algorithm, it is applied to JSS problems and ten benchmark tests with three to eight-objective optimization problems. The proposed algorithm is compared with four state-of-the-art many-objective algorithms (RVEA) [31], IMMOEA [30], NSGA-III [43], Two-Arch [193]), as well as with adaptive reference many-objective algorithms [85].

- *Chapter 6* develops hybrid GP-HH algorithms based on PLS to discover high-quality dispatching rules for JSS problems. An innovative hybridization of global search (GP) with PLS techniques is proposed. In the proposed algorithm, a new selection mechanism for the initial individuals is also proposed. Further, a neighborhood structure is described for GP. Moreover, a sensitivity analysis is performed for finding the total number of generations required to search the solution space extensively and a sufficient number of local search steps utilizing the local search effectively. The key idea of this proposed algorithm is to applied local search effectively and explore the neighborhood of non-dominated dispatching rules. Extensive experiments are performed to understand the effectiveness of the proposed algorithm as compared to the other algorithms (without local search). The experimental studies are carried out using the Taillard static job-shop benchmark set.

- *Chapter 7* summarizes the research goals that have been achieved, followed by the key findings of the thesis. Furthermore, the chapter highlights the potential research areas of future works. .

Chapter 2

Literature Review

This chapter presents the literature for many-objective job shop scheduling (JSS). The literature review starts with a background that covers fundamental concepts, ideas, and algorithms. The literature review then discusses the relevant research works in Section 2.2, ranging from genetic programming based hyper-heuristic (GP-HH) approaches to scheduling problems, many-objective JSS to Pareto local search (PLS). Finally, this chapter concludes with a summary of the literature review in Section 2.3.

2.1 Background

This section covers the basic concepts related to the research works discussed in this thesis. This includes a definition of sequencing and scheduling problems, a definition of machine learning, the conceptual differences between heuristics, meta-heuristics, hyper-heuristics, and dispatching rules. It is then followed by the introduction of evolutionary computation, many-objective optimization, and PLS.

2.1.1 Basic concepts

Sequencing/Scheduling

Sequencing and *scheduling* are a decision-making process that aims to utilize limited resources as effectively as possible. Sequencing and scheduling studies on production planning so that the manufacturing systems generate output with minimal waste in time and money [111].

Sequencing selects a job to process next on a specific machine. Scheduling determines the resource (e.g., machine, work center) to handle each job and the time when the machine begins processing the selected job. Although *sequencing* and *scheduling* are considered two of the crucial decisions for generating a schedule, sometimes, it is not straightforward to utilize them separately due to the complexity of scheduling problems. In our thesis, we use dispatching rules. These rules choose the sequence of jobs at each decision point and begin the process to complete a plan or schedule.

In general, sequencing and scheduling problems can be used to model a wide range of real-world scenarios. JSS is an essential example of scheduling which is briefly discussed in Section 2.1.2. The other types of scheduling problems include flowshop scheduling problems [53] and flexible scheduling problems [165].

Machine learning

Machine learning is a field of computer science that focuses on the area of artificial intelligence [6]. Tom Mitchell [134] gave a "well-defined" mathematical and relational definition for machine learning which is "A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E. In other words, machine learning allows computer programs to improve through experience automatically. Machine learning techniques are categorized into four major categories:

1. **Supervised learning:** The supervised machine learning algorithms handle labeled data with explicit values. The machine learning algorithms in supervised learning environments try to create a function to map inputs to desired outputs which are known in advance. Examples of problems that are handled by supervised machine learning algorithms are classification and regression [172].

2. **Unsupervised learning:** The unsupervised machine learning algorithms deal with the unlabelled data and capture the patterns from a set of data. An example of problems which are handled by unsupervised machine learning algorithms is clustering and neural networks [172].

3. **Semi-supervised learning:** The semi-supervised learning is a machine learning algorithm that carries out both supervised and unsupervised learning.

4. **Reinforcement learning:** The reinforcement machine learning algorithm interacts with a dynamic environment and receives feedback in terms of rewards and penalties [172]. The goal of the learner, in this case, is to maximize the rewards and minimize the penalties [172]. Reinforcement learning is connected to applications for which the machine learning algorithm must make decisions as the learner is not trained to take action and must discover the actions that maximize the rewards (e.g., playing a chess game).

Heuristics/Meta-heuristics/Hyper-heuristics

There are a number of definitions for heuristics, meta-heuristics, and hyper-heuristics. For this thesis, we define heuristics as 'rules-of-thumb. These can be applied to problems directly [172]. These heuristics are useful in cases where it is impractical to find exact solutions to complicated and large-scale problems [133]. For a particular problem, heuristics often incor-

porate existing knowledge about the problem to find good solutions for the problem instances [17]. It means that heuristics are problem-specific [194]. In other words, a heuristic designed for one scheduling problem (a heuristic for minimizing makespan) may not be effective for another (a heuristic for minimizing flowtime) scheduling problem.

Meta-heuristics are defined as problem independent methods that can be applied to solve a wide range of different problems [152]. Meta-heuristics solve problems by directly searching the problem's solution space by incorporating lower level and problem-specific heuristics, which facilitate the search to find good solutions [152]. There are many meta-heuristic techniques developed in the literature and they can be classified into two main categories: (1) local search-based [1] and (2) population-based [152]. The details of these meta-heuristics categories can be seen in Section 2.2.1.

Hyper-heuristics are problem independent techniques that do not directly solve problems, unlike heuristics and meta-heuristics. Hyper-heuristics combine low-level heuristic components and construct a suitable heuristic for any specific problem [17, 20]. The details of these hyper-heuristics categories can be seen in subsection 2.2.1.

Dispatching rules

Most popular heuristics in the literature for solving scheduling problems are referred to as dispatching rules [73, 145]. Dispatching rules are also called priority rules that decide the processing order of the jobs in a queue. The priorities depend on several factors, such as the due date, arrival time, and processing time. The job with the highest priority is processed next to the available machine. Simplicity and flexibility are the two advantages of dispatching rules [165].

Dispatching rules are perhaps the most straightforward method to deal with both static and dynamic JSS problems [20]. In the literature [90, 91], dispatching rules are divided into simple and composite. A simple rule is

a human-made priority function made using various scheduling parameters (processing times, waiting times) such as SPT (shortest-processing-time). On the other hand, a composite dispatching rule is a combination of two or more simple rules. In this thesis, we use genetic programming (GP) to evolve dispatching rules (priority functions), comprising distinctive composite dispatching rules. These rules are applied to JSS based on the machine and job attributes.

2.1.2 JSS Problems

JSS [165] is a significant scheduling problem with a wide range of applications in many industries, such as manufacturing [165] and cloud computing [188]. JSS has been intensively investigated in the literature [62] and has been proven to be *NP-hard* [13]. This subsection covers the definitions and mathematical notations for JSS problems and JSS objective functions. Afterward, active and non-delay scheduling is defined.

JSS: Definition

The general JSS problems are defined based on a set of N jobs and a set of M machines. Each job $j_i,\ 1 \leq i \leq N$ has a sequence of m operations to be performed, i.e. $\{o_i^1, o_i^2 \ldots, o_i^m\}$. Stringent restrictions apply to the processing order's overall operations of a job. Specifically, for any job j_i, the operation o_i^{k+1} cannot start until its previous operation o_i^k has been completed. In other words, operations follow the precedence constraint. Besides, the operations are non-preemptive, i.e., once an operation starts to be processed on a machine, the process cannot be interrupted. Every operation $o_i^k,\ 1 \leq k \leq m$, is further associated with a fixed processing time $p_i^k > 0$ and has to be processed on a specific machine $m_i^k, 1 \leq i \leq M$.

Each job *j* has its pre-determined route and processing time on a specific machine. Classification of JSS depends on machine configurations and the nature of jobs [11]. In *static* JSS, the number of jobs is fixed and

Table 2.1: Notations in JSS

Notation	Definition
$m_1, m_2, m_3 \ldots, m_M$	*Set of M machines*
$j_1, j_2, j_3 \ldots, j_N$	*Set of N jobs*
o_i^k	*k^{th} operation of job i*
D_i	*due date for job i*
R_i	*release time of job i*
w_i	*the weight given to job i*
r_i^k	*ready time of k^{th} operation of job i*
C_i	*completion time of job i*
T_i	*Tardiness of job i*

processing information of all jobs is available such as arrival time and due date. On the other hand, in a *dynamic* environment, jobs continuously arrive and no prior information related to them is available in advance. When an operation has the flexibility of being processed on more than one machine, it is called *flexible* JSS problems. One of the simplest forms of JSS is a flowshop, where all jobs follow the same processing order of operations. Table 2.1 shows the basic definitions and notations in JSS. In this thesis, we will use the GP to evolve dispatching rules for static JSS because many-objective research is comparatively new and the existing research only focus on optimizing single or multi objectives. Furthermore, other forms of JSS environments (dynamic and flexible) are more complex than static JSS environments. This thesis evolved effective dispatching rules for static JSS and understand how optimization approaches can be adapted to improve many-objective JSS algorithms' performance.

Objectives for JSS

A schedule is evaluated by several criteria that can be classified as process-focused performance criteria and customer-focused due date criteria [59].

Process-focused performance criteria pertain to information about the start and end time of jobs and focus on shop performance, such as the utilization of machines. Flowtime and makespan are the most commonly

used performance criteria.

Flowtime is the amount of time a job spends in a shop. Let C_i be the completion time and R_i be the release time of job *i*, then $meanflowtime$ and then $maxflowtime$ [165] are given as:

$$meanflowtime = \frac{1}{N}\sum_{i=1}^{N}(C_i - R_i), \tag{2.1}$$

$$maxflowtime = \max\left\{\frac{1}{N}\sum_{i=1}^{N}(C_1 - R_1), \frac{1}{N}\sum_{i=1}^{N}(C_2 - R_{21}), \ldots \frac{1}{N}\sum_{i=1}^{N}(C_i - R_i)\right\}, \tag{2.2}$$

where N is a number of jobs.

Makespan is the total length of the schedule. In order to optimize the schedule, a minimum makespan (C_{max}) is desired which is given as:

$$C_{max} = \max\{C_1, C_2, \ldots, C_N\} \tag{2.3}$$

Besides performance criteria, some customer-focused criteria are also frequently considered. For example, tardiness is one of the common customer-focused criteria. It measures the amount of time by which the completion time exceeds the due date. Tardiness of job *i* is defined as $T_i = \max\{C_i - D_i, 0\}$. The $TotalTardiness$, $TotalWeightTardiness$, and $MeanWeightTardiness$ are shown below

$$TotalTardiness = \sum_{i=1}^{N}\{\max(C_i - D_i, 0)\}, \tag{2.4}$$

$$= \sum_{i=1}^{N} w_i\{\max(C_i - D_i, 0)\}. \tag{2.5}$$

$$MeanWeightTardiness = \frac{1}{N}\sum_{i=1}^{N} w_i\{\max(C_i - D_i, 0)\}. \tag{2.6}$$

The maximum tardiness $(T_{\max})$ is another difficult objective to minimize due to its sensitivity to the shop condition [147] which is define as:

$$T_{\max} = \max\{T_1, T_2, \ldots, T_N\} \tag{2.7}$$

Different studies have considered different objectives for JSS problems. Banu et al. [23] used a range of artificial intelligence approaches and explored their effectiveness in the JSS environment with different popular objectives such as mean flowtime, maximum flowtime, percentage of tardy jobs, mean tardiness and maximum tardiness.

Active schedules and non-delay schedules

When tackling JSS problems, it is essential to distinguish between active and non-delay scheduling [165]. In a non-delay scheduling, no waiting occurs before the machine starts processing the next job [165, 179].

In other words, a non-delay scheduling algorithm does not allow any delay on the idle machines as long as the waiting queue is not empty. For example, if machine m is available, the scheduling algorithm starts the new job j_i from the waiting queue without any delay.

In the case where no jobs are waiting at the machine m, the non-delay scheduling algorithm will wait until the next jobs arrive and select one of the jobs at the decision point which has the earliest arrival time.

An active schedule allows some reasonable delay (which is no more than the minimal processing time of the waiting jobs) to handle any newly arriving jobs with an urgent due date [165]. Suppose that when a machine m finishes processing a job in an active schedule, a job that has the earliest expected completion time will be processed next by machine m.

The non-delay factor $\alpha \in$ [0,1] determines how many jobs should be considered to be processed next. If a scheduling algorithm has $\alpha = 0$, then the scheduling algorithm is considered a non-delay schedule where the machine will immediately begin processing some operation as soon as there are jobs at the machine m. if $\alpha = 1$, then the scheduling algorithm generates an active schedule for jobs that have arrived at the machine and gives preference to these jobs according to their earliest completion time. Finally, if α is between 0 and 1, then the scheduling algorithm is considered a *hybrid schedule* which has characteristics of both active and non-delay

Table 2.2: Example of a static JSS problem instances (N = 3,M =2)

job	machine sequence	processing time
j_1	m_1, m_2	1,2
j_2	m_2, m_1	2,1
j_3	m_1, m_2	3,1

schedules. The non-delay factor does not go beyond 1.

The following is an example of a static JSS problem instance with the minimization of the makespan objective. This example shows the non-delay and an active schedule in JSS with $\alpha = 0$ and $\alpha = 1$, respectively. Table 2.2 shows job properties (processing order and processing time). Figures 2.1 and 2.2 show the non-delay and active scheduling for a JSS problem, respectively.

In this problem, there are three jobs (j_1, j_2, j_3) and two machines (m_1 and m_2). SPT (where operations with shorter processing times have a higher priority) is used to generate different types of schedules (active or non-delay). For the non-delay schedule, we can see that job j_1 has the shortest processing time than j_3. Therefore, m_1 process j_1 before j_3. Then m_1 begins processing j_3 as soon as j_1 has been completed. On the other hand, in the active schedule, machine m_1 processes j_2 before j_3, as j_2's second operation has a shorter processing time than j_3's first operation, and j_2 arrives at machine m_1 before the expected completion time of j_3. For this problem instance, we can see that the non-delay SPT generates a better solution than active SPT.

2.1.3 Evolutionary computation

Evolutionary computation (EC) is a sub-field of artificial intelligence which focuses on algorithms inspired by the principle of the Darwinian theory of biological evolution [95]. This includes nature-inspired algorithms or population-based systems to deal with various problems. In general, EC techniques are divided into two main categories: (1) evolu-

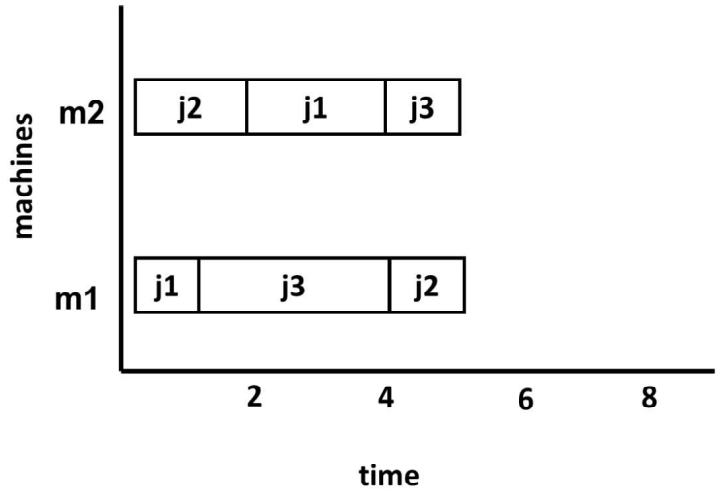


Figure 2.1: The schedule generated by non-delay SPT $\alpha = 0$.

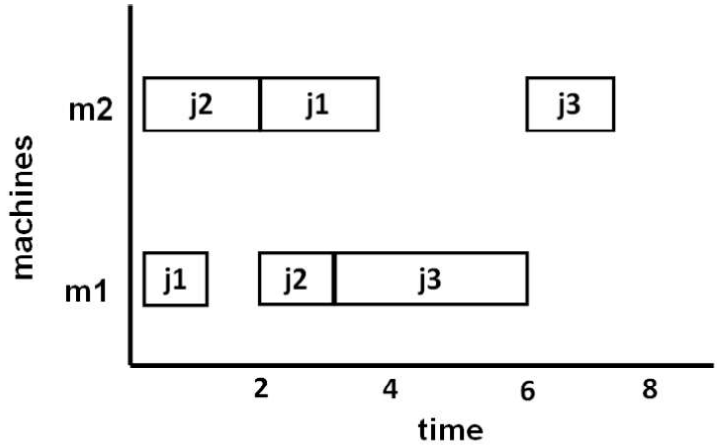


Figure 2.2: The schedule generated by active SPT $\alpha = 1$.

tionary algorithms (EAs) [95] and (2) swarm intelligence (SI) [101]. There is also a third category called miscellaneous algorithms, including the learning classifier system [134].

Evolutionary algorithms (EAs)

EAs are inspired by the principles of natural evolution such as selection, reproduction, crossover, and mutation. In EA, a population of individuals s selected by their fitness value (i.e., individuals' survival depends on their fitness). After the selection, offspring are generated by crossover and mutation. The genetic algorithms (GAs) [69] and GP [105] are the two well-known members of EA that are briefly introduced here.

- **Genetic algorithms (GAs)**: GAs work on finite populations. Each population consists of N chromosomes (solutions) which are typically represented as a fixed-length array [69]. These arrays can represent as bits, integer numbers, or real numbers that carry the information. However, these typical arrays can be decoded to solutions before solving any problem. Then solutions are evaluated according to a specified fitness function and parents are selected to produce offspring through the genetic operators (crossover and mutation). The resulting offspring inherit properties directly from their parents. The offspring are evaluated and a fitter individual is placed in the new population which replaces the weaker solution. The GAs mechanism consists of three phases: evaluation of each solution's fitness, selection of the parent, and applications of mutation and recombination (crossover) operators to the parent for producing offspring. The process is repeated until to have the desired quality of solution or reach the maximum generation.

- **Genetic programming (GP)**: GP is a "search-based automatic programming" technique [105]. GP is an extension of GAs where a population of computer programs is generated with *variable* length. The application of reproduction operators (elitism, crossover, and mutation) produces new programs applied in GP. These programs are then tested against fitness function and those who have higher fitness values are more likely to survive to future generations. In GP,

the programs are traditionally represented as tree structures. Other data structures (linear,graph) have also been employed to construct computer programs. GP is explained in detail in Subsection 2.1.9.

Swarm Intelligence (SI)

SI algorithms are inspired by the collective intelligence of a group of simple agents [12]. In SI, the swarm is an abstract representation of a group of biological organisms that interact with each other to reach a specific goal, e.g., finding food. There are five basic principles of swarm intelligence: proximity, quality, diverse response, stability, and adaptability. A well-known SI algorithm is:

- **Particle swarm optimization (PSO)**: PSO [101] is a swarm intelligence technique where the particles represent a group of organisms such as a flock of birds. The movement of an individual in PSO is influenced by the search direction of the local best (position of the one-particle) and global best (found by all particles). The feedback between the individuals allows PSO to identify high-quality solutions to a given problem quickly.

2.1.4 Multi-objective EAs (MOEAs)

Over the last three decades, MOEAs have been successfully developed to handle various constrained and unconstrained multi-objective optimization problems (MOPs). In general, MOEAs have two ultimate goals with respect to performance which are *convergence* (minimizing the distance to the Pareto-front) and *diversity* (widely and uniformly distributed over the Pareto-front front) [42]. MOEAs have been applied to those problems with two or three objectives. Without loss of generality, in general, MOPs, can be formulated as:

$$min \quad f(\overrightarrow{x}) = \{f_1(\overrightarrow{x}) \dots f_m(\overrightarrow{x})\} : s.t. \quad \overrightarrow{x} \in F \quad f \in Y, \tag{2.8}$$

where F is the complete set of feasible solutions and $x \in F$ is one of the feasible solution, Y $\subset R^m$ is the objective space and f(x) = ($f_1(\overrightarrow{x}) \ldots f_m(\overrightarrow{x})$) $\in$ Y is the objective vector. Here m stands for the number of objectives and m either has values two or three.

Because objectives are mutually conflicting, no single solution can optimize all objectives simultaneously. Instead, we aim to obtain a set of optimal solutions that represent different trade-offs between these objectives. In multi-objective optimization, the comparison of two solutions (x_1 and x_2) is based on the concept of domination [42]. The solution x_1 *dominates* x_2 if and only if

$$\forall i, 1 \leq i \leq m, f_i(x_1) \leq f_i(x_2), \tag{2.9}$$

$$\exists i, f_i(x_1) < f_i(x_2). \tag{2.10}$$

If any other solution does not dominate a solution x_1, then it is called a Pareto-optimal solution (x^*). The set of all Pareto-optimal solutions jointly forms the Pareto-front in the objective space and the Pareto Set (PS) in the decision space. Two very popular MOEAs are:

- **Non-dominated sorting GA-II (NSGA-II)**: The first popular MOEA is NSGA-II, proposed by Deb et al. [44]. NSGA-II is a parameterless elitist strategy which first combines the parent and offspring populations and then performs non-dominated sorting on a combined population followed by the crowding distance.

  Non-dominated sorting uses a dominance rank. The solutions that are not dominated by any other population solutions are assigned the dominance rank of one. Then, the next set of solutions which are only dominated by solutions with a dominance rank of one, is assigned a dominance rank of two, and so forth.

  Crowding distance calculates the average distance of two neighboring solutions. The best solutions (by fitness and spread) are selected through tournament selection for the genetic operators to create the new population.

- **Strength Pareto evolutionary algorithm 2 (SPEA2)**: The second popular MOEA is SPEA2 proposed by Zitzler et al. [211]. Like NSGA-II, is also an elitist method. SPEA2 has a separate fixed-size archive where the population is updated at each generation with non-dominated individuals. The archive can eliminate individuals if the non-dominated fronts exceed the fixed size. Individuals' fitness is a combination of a dominance strength (which is the number of solutions in the archive and current population dominated by the individual) and density information. There are many other MOEAs in the literature, such as MOEA/D [205]. This thesis evolves trade-off (non-dominated) dispatching rules for many-objective JSS problems, therefore, we focus on Pareto dominance based MOEAs such as NSGA-II and SPEA2.

- **MOEA based on decomposition (MOEA/D)**: MOEA/D [205] decomposes MOP to single-objective problems by employing three possible decomposition functions [205]. These functions are the weighted sum function, the Chebyshev function, and the penalty-based boundary intersection function, which decompose the multiple-objective problems into a set of scalarizing subproblems. It maintains population diversity by a predefined set of weight vectors. Several variants of MOEA/D have been proposed for enhancing the selection strategy of each sub-problem [205].

2.1.5 Many-objective optimization (MaOPs)

Over the years, evolutionary multi-objective (EMO) algorithms have two performance-centric considerations: *convergence* (minimizing the distance to the Pareto-front) and *diversity* (widely and uniformly distributed over the Pareto-front) on a problem with two or three conflicting optimization objectives [42]. If MOPs have four or more than four objectives (i.e., $m \geq 4$), then it is called MaOPs [116].

Many real-world problems have been naturally defined as MaOPs [43].

Prominent examples include water resource engineering [138], nurse rostering [180], car control system design [140], and air traffic control [65]. In recent years, MaOPs have drawn great attention in the EMO community, intending to tackle various new challenges caused by handling many concurrent objectives.

MaOPs pose a number of challenges to Pareto-based MOEAs [84]. First, the number of non-dominated solutions increases at an exponential scale with an increasing number of objectives. Therefore, most Pareto-based MOEAs, such as NSGA-II and SPEA2, cannot provide sufficient selection pressure towards the Pareto-front. Second, commonly used diversity-preservation operators such as crowding distance [44] and clustering [211]. These operators become computationally expensive when they used in MaOPs [84].

Categories of many-objective optimization algorithms

Many-objective optimization approaches are roughly classified into the three classes: (1) dominance-based approaches, (2) decomposition-based approaches, and (3) indicator-based approaches.

- **Dominance-based approaches** solve the MaOPs by enhancing the convergence pressure towards the Pareto-front. Since most of the MOEAs lose their selection pressure toward the Pareto-frontbecause of Pareto-dominance's inability to distinguish solutions. To solve this issue, most MOEAs use the convergence enhancement which modifies the dominance relationship to increase the Pareto-frontselection pressure. Several dominance modification approaches have been proposed, such as grid-dom [199] and α-dom.[77]. Many other algorithms are also used to combine the additional convergence-related metrics with the Pareto-dominance-based criterion. An example is an indicator-based preference combined with the dominance criterion for speeding up the convergence

of NSGA-II [27]. One of the recently proposed algorithms, the knee point-driven EA (KnEA) [208], enhances the convergence pressure by assigning the higher selection to the non-dominated fronts' knee points in the current population. Dominance-based approaches enlarge the dominating area of the non-dominated solution but make diversity maintenance more difficult [116].

- **Decomposition-based approaches** convert a complex MOP to a number of sub-problems and simultaneously solve them. They are another promising method for MaOPs. This class of MaOPs is further divided into two other types of decomposition-based approaches.

  The first type of decomposition-based approach decomposes multiple objectives into a single objective using scalarizing functions and weighting vectors. These approaches use the aggregated criteria to distinguish solutions. Multiple single-objective Pareto sampling [72] and MOEAs based on decomposition (MOEA/D) [205] are the two famous aggregation-based algorithms. These approaches have two major challenges: selecting weighting vectors and diversity maintenance subject to the limitation of adopted scalarizing functions. Several variants of MOEA/D provide a better balance between convergence and diversity by enhancing the selection strategy for each sub-problem [205].

  In the second type of decomposition-based approach, an MOP is decomposed into other sub-MOPs. For instance, the research studies in [31, 43] partition the whole Pareto-front into a Pareto-front subset, and each sub-Pareto-front can be considered a sub-problem. Another MOEA that mainly falls under this category is NSGA-III [43] that employs a set of uniformly distributed reference points to preserve the diversity of the candidate solutions. The results on several test problems and practical problems have shown NSGA-III's useful-

ness in solving three- to 15-objective constrained and unconstrained optimization problems [43], [86].

The reference vector-guided EA (RVEA) [31] is one of the prominent decomposition-based approaches. RVEA uses reference vectors to decompose the MOPs into a number of single-objective subproblems via objective function aggregations and use user preferences to target preferred subspaces of the whole Pareto-front.

- **Indicator-based approaches** Indicator-based approaches use indicator values to guide the search process. An example of indicator-based approaches is the S-metric selection-based evolutionary multi-objective algorithm [132] and a dynamic neighborhood MOEA based on HV indicator [117, 18]. These approaches do not have dominance-based MOEAs issue for solving MaOPs. Unfortunately, the calculation of the performance indicator (HV) becomes computationally expensive especially in the case of many-objective [18].

A few other approaches do not fall into any of the above three main categories. These approaches include the diversity-based approaches, preference-based approaches, and dimensionality reduction-based approaches.

The diversity-based approaches are used in the selection of a solution when the primary selection criterion fails. Diversity-based approaches try to minimize the adverse impact of diversity in MaOPs. Many diversity measures have been proposed, such as the shift density estimator [119] and the grid-based neighborhood niching [199].

The Preference-based approaches select the biased region of search space with the help of user preference. This approach can be further categorized into three classes, a priori, interactive, and posterior methods, according to the timing when users give their preference (before, during, or after the optimization).

Dimensionality reduction-based approaches try to reduce dimensionality

by reducing the number of objectives. They might lose some important information as a result of collectively handling several relevant objectives.

It is expected from population-based optimization algorithms to convergence its population near the Pareto-optimal front and distributed uniformly around the entire front. In this thesis, we evolve dispatching rules which will help decision-makers by providing them with potential trade-offs among different objectives. In this scenario, indicator-based approaches can not be a good choice because they have high computational cost. Further, few indicators-based approaches are not strictly monotonic with Pareto dominance and might lead to performance degradation. Next, dominance-based approaches modify the Pareto-dominance idea by enlarging the non-dominated solutions' dominating area and improving the convergence criteria. However, the drive toward a more aggressive selection pressure seems to make diversity maintenance more difficult in a dominance-based MOEAs. In contrast, decomposition-based approaches distribute reference points evenly in the whole objective space and manage the candidate solutions' diversity, eventually contributing to enhanced convergence of the algorithm and finding well-converge and widely distributed sets of Pareto-optimal solutions.

Non-dominated sorting GA-III (NSGA-III)

The basic framework of NSGA-III is similar to the NSGA-II algorithm. The difference between NSGA-II and NSGA-III is the maintenance of diversity among population members. NSGA-II uses the crowding distance measure for selecting a well-distributed set of Pareto-optimal solutions. In contrast, NSGA-III is aided by the supplied well-spread reference points (Z_r) for selecting the Pareto-optimal solutions.

In NSGA-III, the parent population (P_g) of size N is randomly generated in the specified problem domain which is then followed by the selection of binary tournament, application of crossover and mutation operators to P_g. Then, an offspring population Q_g of size N is created. This is

then followed by the combination of P_g and Q_g which is called R_g ($P_g \cup Q_g$) of size 2N.

The combined population (R_g) is sorted according to different non-domination ranks $(F1, F2, \ldots F_k)$. Each rank $(F1, F2, \ldots F_k)$ is selected one at a time and stored in a new population S_g of size N. After the non-dominated sorting, members of all the acceptable levels and the last level (l_{th} level) are stored in S_g. The l_{th} level is usually accepted partially. The niche-preservation operator selects the members of the well-distributed solution of F_l.

The supplied reference points (Z_r) are used to select the remaining members in NSGA-III. The reference points are generated systematically [39] and place on the normalized hyper-plan. Objective values are also normalized so that they have an identical range of reference points. Thereafter, the perpendicular distance between a member in S_g and each of the reference point is calculated. The member is then associated with the reference point having the smallest perpendicular distance. Next, the niche counts ρ for each reference point is calculated. The reference point having the minimum associated solution is identified and the associated members of the last rank F_l enter in the final population. The selected reference point's niche count is increased by one and the procedure is repeated to fill up the population (P_{g+1}) of a new generation.

Issues in handling many-objective optimization

The literature has discussed that EMO algorithms may face the difficulties of scalability and diversity measures [43]. In this thesis, we focus on scalability and diversity issues which are briefly discussed below:

- **Scalability** (in terms of a number of objectives) is considered as one of the major challenges where MOEAs cannot provide sufficient pressure toward the Pareto-front. Conventional Pareto-based EMO algorithms encounter difficulties when the number of objectives in-

creases [43]. This is because all the candidate solutions become non-dominated and the Pareto-based selection criterion cannot clearly distinguish among these candidate solutions. Therefore, NSGA-II and it is contemporary fail to handle more than three objectives problems.

- **Diversity Maintenance** plays an important role in the selection of the fittest solutions when the primary selection criterion (convergence) cannot readily find good solutions in EMO. For the sake of enhancing diversity, EMO promotes the evolution of uniformly distributed candidate solutions [43]. However, in the case of a high dimension, it is quite challenging to maintain solution diversity because solutions distribute very sparsely in a high-dimensional space. Moreover, diversity-preservation operators of EMO such as crowding distance [44] and clustering [211] cannot strengthen the selection pressure toward the Pareto-front and also suffers from high computational cost. Diversity maintenance can be controlled by providing multiple predefined reference points. Most of the algorithms [31, 43] employ a predefined set of uniformly distributed reference points to encourage the generation of wide distributed solutions on the whole objective space.

2.1.6 MaOPs for irregular Pareto-front

It has been verified that the many-objective algorithms' performance is highly dependent on the curvature of the Pareto-front [80, 118]. Hence many algorithms that use uniformly distributed reference points can perform very well on the MaOPs when Pareto-front has regular shapes [86] such as DTLZ-1 [43]. RVEA [31] and NSGA-III [43] show remarkable results, especially when Pareto-front is uniform. However, their performance deteriorates considerably when these algorithms are applied to irregular, non-uniform, and discontinuous Pareto-front [85, 86]. Recently,

Ishibuchi et al. [80] found that the reference-point-based algorithms can perform effectively if the distribution of reference points is consistent with the distribution of Pareto-optimal solutions. Motivated by this discovery, several algorithms such as A-NSGA-III [85] and RVEA* (extension of RVEA) [31] have been proposed which adjust reference points adaptively.

These reference-point-adaptation algorithms mainly learn the distribution of the candidate solutions and adjust the reference points accordingly. The reference points based EA for many-objective optimization (REPA) [123] is an extension of NSGA-III which generates a set of reference points according to the current population. In the environmental selection process, the fittest individuals are selected by calculating the Euclidean distance between the reference points and individuals.

A-NSGA-III [85] is a notable improvement in NSGA-III. This extension of NSGA-III is designed to relocate the reference points by two major operations dynamically: (1) inclusion and (2) exclusion. In the inclusion procedure, multiple reference points are added around the crowded reference points with a high niche count. These reference points are generated by a centroid method where the original points are considered as a centroid. However, in A-NSGA-III, it is not easy to introduce new points around the vertices of simplex because this violates the positive quadrant's condition [85]. Due to this reason, A-NSGA-III cannot avoid useless points completely and fails to guide the evolution of a well-distributed set of Pareto-optimal solutions.

RVEA* [31] is a well-known state-of-the-art algorithm for reference point adaptation. RVEA* uses the replacement approach instead of the deletion and addition operations. RVEA* first partitions the current population into different subspaces. Next, reference points around these empty subspaces are replaced by the new non-empty subspace locations based on objective values' ideal and nadir points.

Generation of reference points

Without loss of generality, this thesis's reference points are positive (they are all in the first quadrant). In this approach, a set of uniformly distributed reference points (z) is generated by the canonical simplex-lattice design method [35] as

$$\begin{cases} z_k^i \in \left\{ \frac{0}{H}, \frac{1}{H}, \ldots, \frac{H}{H} \right\}, & \sum_{i=1}^{m} z_k^i = 1 \\ \vec{z}_k = (z_k^1, z_k^2, \ldots, z_k^m), & k = 1, 2, \ldots, K \end{cases} \tag{2.11}$$

where $i = 1, 2, \ldots, K$ are the K uniformly distributed points, m is the number of objectives, and H is a positive integer used in the simplex-lattice design.

2.1.7 Model-based EMO

Traditional EAs such as NSGA-II and SPEA2 generate new individuals for the next generation using genetic operators (e.g., crossover and mutation) [42]. These EAs generate solutions without utilizing any problem-specific knowledge. Unlike these algorithms, model-based evolutionary algorithms (MBEAs) are designed to learn desired features or structure problems explicitly.

Recently, many MBEAs have been proposed and one of the recent surveys in [29] classified MBEAs into three categories: (i) Estimation of Distribution Algorithms (EDAs) [109], (ii) Inverse Modelling [30], and (iii) Surrogate Modelling [93].

EDA is the most popular category of MBEA and it mainly focuses on estimating the solution distribution. EDA estimates the distribution of the Pareto-optimal solutions by training and sampling in the decision space [109]. In EDA, a machine learning model such as a regression model, is iteratively refined to estimate the distribution of solutions as the evolution proceeds.

In the earlier research of EDA, conditional probabilistic modeling has been used which enables piece-wise interaction between candidate solutions. The research studies in [41, 161] show an elementary work of pairwise intersection in the linear model. On the other hand, complex interactions among the solutions can also be handled through the Bayesian network as a multivariate model [160]. The Bayesian network is also used to build a multi-objective model and to balance convergence and diversity. For example, Bayesian multi-objective optimization algorithm $(BMOA)$ [110] introduces the new selection operator ϵ which is based on a ϵ-archive that keeps track of a minimal set of solutions that dominates all other solutions generated so far. One of the research studies in [151] proposed a Voronoi-based EDA which helps to put the candidate solutions on different fronts.

The multi-objective EDAs can effectively preserve the diversity of the population is shown in [15, 186, 35]. Multi-objective EDA usually builds the model in a decision space but the study in [30, 36] suggests directly controlling the distribution of the solutions through an inverse model. Notably, the Gaussian process-based inverse model aims to construct the Gaussian process model that maps the optimal solutions from the objective space to the decision space [30]. This model is subsequently utilized to sample candidate solutions with the help of a reproduction operator.

Similar to the related works summarized above, in this thesis, we will adopt a modeling technique to tackle one of the challenges of MaOPs.

2.1.8 Gaussian process modelling

A Gaussian process (GaP) is a stochastic process which has a collection of random variables and can be seen as a generalization of multivariate Gaussian distribution in the function space [170].

In particular, if any finite sub-collection of random variables $r(x) : x \in X$ are drawn from a GaP, these variables are fully specified by a

mean function $\mu(x)$ and a *covariance function* $K(x_i, x_j)$. The following equation can denote the GaP::

$$r(x) \sim GaP(\mu(X), K(x_i, x_j)), \tag{2.12}$$

where the covariance function $K(x_i, x_i)$ measures the correlation between any of the two arbitrary x_i and x_j which is expressed as:

$$\mathbf{K} = \begin{bmatrix} K(x_1, x_1) & \cdots & K(x_1, x_j) \\ \vdots & \ddots & \vdots \\ K(x_i, x_1) & \cdots & K(x_i, x_j) \end{bmatrix}. \tag{2.13}$$

One of the most commonly used covariance functions is the squared exponential in which the correlation of x_i and x_j is measured as a function of their distance:

$$K(x_i, x_j) = \sigma_f^2 \exp\left(-\sum_{k=1}^{n} \theta_k \left| x_i^k - x_j^k \right|^2 \right), \tag{2.14}$$

where σ_f^2 and θ_k are hyperparameters of the covariance functions.

Assuming the regression model in equation (2.15) which shows the relationship between x (input vector) and y (output vector), we have:

$$y(x_i) = r(x_i) + \epsilon(x_i), \tag{2.15}$$

where $\epsilon(x) \sim N(0, \sigma^2)$ and $i = 1, 2 \cdots, m$.

Training of GaP focuses on the mean function $\mu(x)$ and the covariance function $K(x_i, x_j)$. The prior distribution of the Gaussian process regression model is shown below as a latent function.

$$r(\cdot) \sim GaP(0, K(\cdot, \cdot)), \tag{2.16}$$

In order to calculate this posterior distribution of the model, a distribution over the hyperparameters $p(\theta|y, X)$ needs to be defined. In general, log-likelihood could be used for estimating hyperparameter values which is explained as:

$$logp(\theta|y,X)\left(\theta\right)=-\frac{1}{2}\left(log|K|+y^{\top}K^{-1}y+N\log\sigma^{2}\right), \tag{2.17}$$

Based on the training set X, a covariance matrix K of size $N \times N$ can be calculated. With respect to any given input vector x^*, the corresponding predicted output y^* can be obtained by a Gaussian probability distribution with the mean and covariance given as:

$$\begin{aligned}\mu(y^{*}) =&K_{*}^{\top}\left(K+(\sigma_{n})^{2}I\right)^{-1}y^{*},\\ \left(\sigma(y^{*})\right)^{2} =&K_{**}-K_{*}^{\top}\left(K+(\sigma_{n})^{2}I\right)^{-1}K_{*}.\end{aligned} \tag{2.18}$$

where K_* =$[K\left(x_1,x^*\right),....,K\left(x_N,x^*\right)]$ is the $N \times 1$ vector of covariances between the test and the training cases, and K_{**} is the covariance between the test input itself.

2.1.9 Genetic programming (GP)

GP [105] is an EC-based meta-heuristic search method which can evolve computer programs to perform specific tasks automatically. GP individuals are represented as variable-length data structures. Different data structures have been used for GP representation, such as tree-based [105], graph-based [120] and linear-based [203] representations. The tree-based representation is flexible in nature because the size of the evolved tress can vary from minimum depth to maximum depth [40]. This means GP has the potential to cover a larger search space of potential functions GAs (because of fixed-length representation).

Due to its flexibility, GP can be applied to various machine learning and optimization problems, including numerous combinatorial optimization problems such as JSS. Tree-based GP has been widely used in JSS literature to evolve dispatching rules [145, 147]. This thesis will focus on using tree-based GP for evolving dispatching rules to construct schedules.

Representation

In GP, solutions are normally represented as Lisp S-expression. Tree-based GP is represented by a tree structure. Trees are constructed by internal nodes and leaf nodes. In general, leaf nodes are called terminals and they consist of variables and constant numbers. On the other hand, internal nodes are functions defined by user in terms of different operators such as arithmetic operators $\{+, -, *, /\}$ and logical operators {OR, AND, XOR} [178]. An example of a tree-based GP is given in Figure 2.3. In the GP tree, the selection of a terminal set and function set can be problem-specific. For example, ready time $\{RJ\}$, number of remaining operations $\{RO\}$, work remaining of job $\{RT\}$, operation processing time $\{PR\}$, weight $\{w\}$, due date $\{DD\}$, and machine ready time $\{RM\}$ are commonly considered job shop attributes [145].

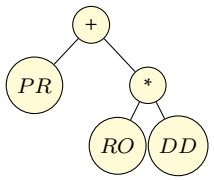


Figure 2.3: An example of a GP :$PR + (RO \times DD)$).

Initialization

The initial individuals are usually generated by two earliest initialization methods: full and grow methods [105]. In the full method, nodes are selected from the function set and appended to the tree until the tree's depth reaches a predefined maximum depth. After that, the members of the terminal set are used as leaf nodes. In the full initialization procedure, all the terminals are located at a maximum depth of the trees.

In the grow approach, the leaf nodes do not need to be at maximum depth. Therefore, the nodes are randomly picked from both the terminal

set and the function set. The grow method generates trees with more variety in shape and size as compare to the full method.

Another initialization method is the ramped half-and-half method [104]. In order to generate a wide variety of trees of various depths, lengths, and shapes, the full and grow methods are often combined. In the ramped half-and-half method, half of the trees are created by the full method and the other half are constructed by the grow method.

Evaluation

A driving force of GP is the fitness function. It helps guide the search to find good individuals from the population-based on the fitness values of those individuals. The uses of the fitness function depend on the problem domain. For example, consider a JSS problem to minimize tardiness. In this problem, the fitness value of each individual is calculated as follows. First, the individual from a tree-based GP population is applied to training instances as a priority dispatching rule and generates solutions. Afterward, the fitness function calculates the mean tardiness of the solutions. If an individual A_1 has a fitness value lower (in a minimization problem) than an individual A_2, A_1 is considered better than individual A_2.

Selection

In a natural evolution, next-generation is produced by selecting the fittest individuals from the current population. GP follows a similar approach and after the evaluation phase, fitness values are used to select individuals to generate offsprings for the next generation. It should be noted that better individuals are more likely to survive in the evolutionary process. Two popular selection methods are the *roulette wheel selection* and the *tournament selection*.

In the roulette wheel selection [105], an individual is randomly selected based on the probability determined by its fitness value. Thus, an indi-

vidual having a higher fitness value will have a higher chance of being selected.

In the tournament selection method [105], a number of individuals are sampled randomly from the entire population into the tournament. The best individual from the tournament is then selected and is placed into the mating pool for mating by genetic operators. The mutation uses one tournament because of the requirement of a single parent individual. The Crossover uses two tournaments because of the requirement of two-parent individuals.

Genetic operators

There are three main genetic operators used in GP which are crossover, mutation, and reproduction. They together produce new programs from the selected individuals.

The crossover is applied to two selected individuals known as the parents. A subtree is then randomly picked from each parent. Next, following the subtree crossover method, the selected subtrees are exchanged and two offsprings will be created as a result. A node is randomly picked from each parent who is identified as a crossover point. This is known as a subtree crossover, where two offsprings are produced by swapping the subtree from the two parents. Figure 2.4 represents an example of a subtree crossover.

The mutation operator selects only one parent to create an offspring. The most common mutation approach is the subtree mutation, in which a random node is chosen as a mutation point from a selected parent and the subtree rooted from the node is removed. After that, a new subtree is grown from that mutation point. Figure 2.5 shows a subtree mutation example.

In the case of reproduction, all the selected individuals are copied to the new population. This operator ensures the good individual will survive during the evolutionary process.

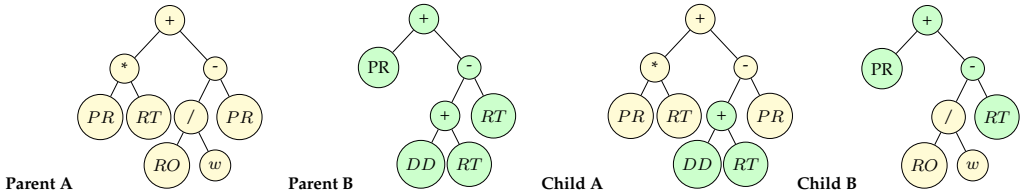


Figure 2.4: An example of a crossover operation in GP.

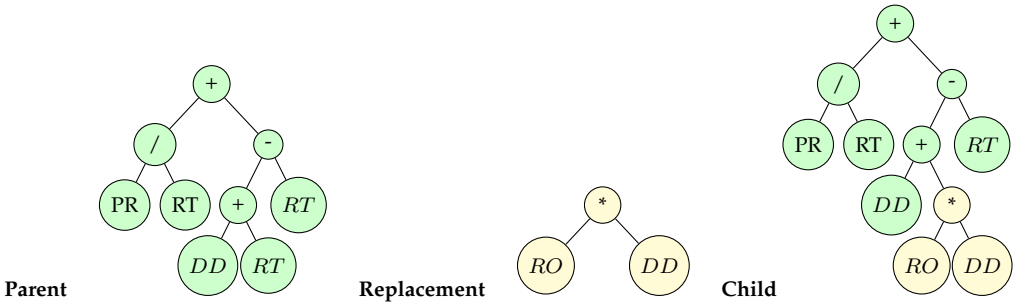


Figure 2.5: An example of a mutation operation in GP.

2.1.10 Basic GP algorithm

Algorithm 1 shows a basic GP algorithm, which aims to find the best performing individual program from the evolved programs. The algorithm starts with the initialization of a randomly generated population of a size specified, S. Then, evaluation, selection, and genetic operations (discussed above) are carried out. Every individual Δ_i in the population of programs P is evaluated using a pre-determined fitness function. Suppose the evaluated program is better (has smaller fitness value since we focus on minimization problems in this research) than the best program Δ^*. In that case, it will be assigned to the best program found so far Δ^* and the best fitness value $fitness(\Delta^*)$ is also updated. Once all the best individuals in the population are evaluated, new individuals for the next generation are generated by applying mutation, crossover, and elitism to individuals selected using the selection mechanism. The algorithm repeats the evolutionary process until the maximum number of generations is attained and returns the best found rule Δ^*.

Algorithm 1: Basic GP algorithm.

Input : problem data
Output: the best evolved program Δ^*

randomly initialize the population $P \leftarrow \{\Delta_1, \ldots, \Delta_S\}$;
set $\Delta^* \leftarrow null$ (best rule);
set $fitness(\Delta^*) \leftarrow +\infty$;
$generation \leftarrow 0$;
while $generation < maxGeneration$ **do**
    foreach $\Delta_i \in P$ **do**
        evaluate and assign fitness to Δ_i;
    end
    foreach $\Delta_i \in P$ **do**
        if $fitness(\Delta_i) < fitness(\Delta^*)$ **then**
            $\Delta^* \leftarrow \Delta_i$;
            $fitness(\Delta^*) \leftarrow fitness(\Delta_i)$;
        end
    end
    set $P^{new} \leftarrow \emptyset$;
    while $|P^{new}| < S$ **do**
        $\Delta^{new} \leftarrow$ apply genetic operators crossover, mutation and reproduction to selected programs from P;
        $P^{new} \leftarrow P^{new} \cup \Delta^{new}$;
    end
    $P \leftarrow P^{new}$;
    $generation \leftarrow generation + 1$;
end
return *the best individual* Δ^*;

2.1.11 Local search

Local search [38] is a meta-heuristic approach to improves the search effectiveness and quality of individuals. The local search method is typically

presented as an iterative process that starts with an initial solution generated randomly or by some constructive heuristic. The method iteratively improves the current solution by moving to better neighboring solutions defined by the neighborhood function. If the current solution s_0 found a better neighborhood solution s_1, it replaces the current solution s_0 and the search is continued from s_1. If no better solution is found, the algorithm terminates at a local minimum. The exploration strategies in local search determine the number of the neighborhood to be explored [149]. The most often used strategies for exploration in the literature are the *first-improvement* and the *best-improvement* [79]. In the first-improvement strategy, the first best neighboring solution is selected. The best-improvement strategy explores the neighborhood entirely and returns the best neighboring solution.

The combinatorial optimization problem's solution is represented by discrete structures like graphs, sequences, and genetic programs The neighborhood function introduces suitable perturbation operators tailored for the corresponding representations to generate 'neighboring' solutions via swapping, moving, or replacement. In subsection 2.2.4, we will discuss the works related to local search methods applied to JSS problems.

2.1.12 Pareto local search (PLS)

PLS can be considered as a direct extension of local search from single-objective problems to the multi-objective case. Generally, PLS maintains the Pareto archive of non-dominated solutions.

Algorithm 2 illustrates the PLS framework. The input to PLS is an initial set of solutions S_0 that are mutually non-dominated. The solutions in S_0 are initially marked as unexplored (line 2). PLS updates an archive of non dominated solutions S which is initially equal to S_0. PLS starts with an initial Pareto archive and then solution s is chosen randomly among all unexplored ones in the archive. Next, the neighborhood of solution of s,

Algorithm 2: Pseudo-code for Pareto Local Seach

```
   Input  : An initial set of solutions S_0
   Output: S
1  explored(s)← FALSE ∀_s ∈ S_0;
2  S←S_0;
3  while S_0! = ∅ do
4      s ← select randomly a solution from S_0;
5      foreach s' ∈ N(s) do
6          if s' non-dominated with respect to the solutions in S then
7              explored(s')← FALSE;
8              S ← Update(S, s');
9          end
10     end
11     explored(s) ← TRUE;
12     S_0 ← {s ∈ S|explored(s) = False} ;
13 end
14 return the best individual Δ*;
```

$N(s)$ is explored. In PLS, the acceptance of a new neighborhood solution is based on the dominance relation. The procedure $Update$ (line 7) adds an $s' \in N(s)$ candidate solution to the archive if s' is not dominated by any solution in the archive and removes all the archive solutions that become dominated by s'. Whenever a new neighbor solution s_1 not dominated by the current solution is found, the Pareto archive needs to be updated. The s_1 is accepted if it is not dominated by all solutions in the archive and all dominated solutions by $s1$ are removed from the archive. Furthermore, PLS has a natural stopping condition when the neighborhoods of all solutions in the archive have been explored (line 11). Liefooghe et al. [121] generalize PLS algorithms into several categories based on the different ways of defining its basic components: (1) initial solutions strategy for neighborhood scanning, (2) exploring the neighbors of a non-dominated solution, fully vs. partially, (3) archiving method, e.g., bounded vs. unbounded, and (4) stopping conditions.

Selection

The selection strategy selects the solutions of the current archive whose neighborhood will be explored. Either all of the solutions are selected from the archive or only a subset of solutions may be selected. The subset of solutions is selected either uniformly at random or according to the fitness of solutions.

Exploration

Explorations strategies control the size of the neighborhood for exploration. PLS can either explore the neighborhood entirely (fully) or only partially until their termination criterion is met.

Archiving

Two types of archiving strategies are considered. In the first strategy, the archive can have a set of all non-dominated solutions. In the second strategy, the archive can have a subset of non-dominated solutions. The second strategy limits the archive's size using some criteria such as diversity criteria (crowding distance).

Termination criteria

A common termination criterion is the one when all the solutions are fully explored. Otherwise, other popular termination criteria consider the time spent by the algorithm, the number of iterations, and a number of iterations without improvement [50]. Several PLS-based methods have also been developed for solving multi-objective JSS [14].

2.2 Related work

This section covers the related works that have been carried out in various fields of research that are relevant to the research goals and objectives. First, we cover the exact optimisation, heuristic, meta-heuristic, and the GP-HH approaches that have been proposed in scheduling research. Afterward, we cover the multi-objective and many-objective JSS. Finally, we cover local search and PLS that have been applied to JSS problems.

2.2.1 JSS techniques

This section covers the exact optimisation, heuristic and meta-heuristic approaches to JSS techniques.

Exact optimisation techniques

Exact optimization techniques such as Jackson's algorithm [83] can solve two-machine JSS problems to minimize makespan. Garey et al. [53] showed that static makespan minimization JSS problems with the number of machines ($M > 2$) are NP-hard. This means that no algorithm can spend polynomial time in the worst-case scenario ($M > 2$) to minimize the makespan in a job shop. Researchers have suggested exhaustive search techniques to handle more difficult JSS problems. Branch-and-bound [107] is one of the search techniques which have been used extensively in the literature [5, 7, 19, 24, 25]. Carlier and Pinson [25] have proposed a notable branch-and-bound technique where they were able to find an optimal solution for a JSS problem instances proposed by Muth and Thompson [139] with $N = 10$ jobs, $M = 10$ machines, and 10 operations per job. Branch-and-bound techniques are covered in detail in a survey paper by Potts and Strusevich [166].

Dynamic programming is an exact optimization technique that has been applied to static JSS problems [60, 112, 165]. Dynamic programming

approaches divide a JSS problem instance into sub-problems and attempt to solve each sub-problem separately. Lawler and Moore [112] proposed a technique of applying dynamic programming to a single machine scheduling problem with total weighted tardiness minimization. They suggested methods of extending it to parallel and two machine flowshop problems. A more recent dynamic programming approach to JSS with makespan minimization has been proposed by Gromicho et al. [60] in 2012, where they adopt an approach proposed by Held and Karp [64] for the traveling salesman problem (TSP). Their analysis shows that the dynamic programming approach can generate optimal solutions for moderate benchmark instances where problem instances have up to $N = 10$ jobs and $M = 5$ machines.

Although exact mathematical optimization techniques guarantee an optimal solution for a JSS problem instance, they generally take too long to generate optimal solutions for problem instances that have more than 10 to 15 jobs [60]. In addition, exact mathematical optimization techniques are not suitable for more complex problems because they generally take too long to generate optimal solutions such as many-objective JSS [5].

Heuristic techniques

Heuristic approaches (e.g., shifting bottleneck heuristic) have been applied to large static JSS problems, where problem instances have up to $N = 8700$ jobs and $M = 9$ machines [154]. They have also been applied to dynamic JSS problems where it is not appropriate to use exact optimization techniques as a schedule may not be optimal as new unforeseen events occur during processing [181]. Examples of heuristics are small and simple dispatching rules such as Shortest Processing Time (SPT), First In First Out (FIFO), and Earliest Due Date (EDD) [165]. The EDD rule prioritizes jobs based on the earliest due and time. Vepsalainen et al. [191] covers multiple composite dispatching rule heuristics for JSS problems with Total Weight Tardiness (TWT) minimization objective, in-

cluding the COVERT (cost over time) and ATC (apparent tardiness cost) priority-based dispatching rules. Gupta et al. [61] use dispatching rules in semiconductor manufacturing. They show that dispatching heuristics provides schedules quickly that are easy to understand, easy to apply, and require relatively short computation time. The primary disadvantage of manually dispatching rules is that these cannot hope for optimal solutions for all performance measures in the dynamic job shop [147].

Jayamohan et al. [90] compares various dispatching rules which have been proposed in the literature, such as rules proposed by Holthaus et al. [70] and also introduced several new dispatching rules. They performed a computational analysis of the rules on dynamic JSS problems for minimizing various aspects of flowtime and tardiness. They further extended the comparative study to incorporate weighted COVERT [90] and weighted ATC rules [191] and propose new dispatching rules. Nguyen et al. [145] proposed three different representations of dispatching rules and suggested that the representation that integrates the system and machine attributes can improve evolved rules' quality.

Holthaus et al. [70] proposed the $2\text{PT} + \text{WINQ} + \text{NPT}$ priority rule and variations of the rule for the dynamic JSS problems with flowtime and tardiness minimization objectives. PT denotes the processing time for the job's operation. WINQ denotes the work-in-next-queue, which is the sum of processing time of other jobs waiting at the *next* machine that the job will be processed on. NPT denotes the processing time of the job's operation on the next machine, it needs to be processed or 0 otherwise. They show that $2\text{PT} + \text{WINQ} + \text{NPT}$ consistently outperformed the other benchmark dispatching rules, such as the RR rule (Raghu and Rajendran's rule) [168] for minimizing mean tardiness for dynamic JSS problem instances. [63] considers multiple shop scenarios in a dynamic shop environment and apply Gaussian process regression to switch dispatching rules between EDD, MOD, 2PTPlusWinqPlusNPT [169].

A more complex example of a heuristic approach is the shifting bot-

tleneck (SB) heuristic which was proposed by Adams et al. [3] in 1988 for static JSS problem with makespan minimization objective. Adams et al. showed that the heuristic approach can find high-quality optimal solutions for problem instances with up to $N = 15$ jobs and $M = 15$ machines.

Meta-heuristic techniques

The research on meta-heuristics for JSS problems has also been extensively studied in the literature where both local search-based techniques and population-based techniques have been effectively applied to different JSS problems.

- **Local search** Local search based methods such as simulated annealing [190] and Tabu search [162] have shown very promising results in JSS. These methods begin with a complete schedule and try to improve the schedule by developing efficient and effective neighborhood structures, mainly based on the concept of critical paths and critical blocks. They also have diversifying strategies to escape from local optima. Many of the research work in [3, 7, 139] applied tabu search to various benchmark JSS problems and showed that Tabu search approaches can found optimal solutions for the static JSS problem instances. Like Tabu search, simulated annealing [190] approaches have been applied effectively to JSS problems in the literature. Kreipl [106] proposed a large step random walk (LSRW) method. This is considered one of the best local search methods for dealing with JSS problems with tardiness objectives. The detailed information of local search in JSS can be found in subsection 2.2.4.

- **Population-based techniques** Cheng et al. [28] surveyed the GAs to JSS problems, showing that GAs consists of a population of individuals represented by a fixed-length chromosome. One of the notable examples of a GA to JSS problems with tardiness objective is proposed by Zhou et al. [206]. The GA algorithm is hybridized with

the existing heuristics for JSS. The experimental result showed that the hybrid GA outperforms pure GA approaches and significantly reduces the computation time required to generate a solution. A hybrid GA method and neighborhood local search for JSS to minimize makespan was proposed in [173]. In this method, the random keys were used for chromosome representation which represents the priorities of operations.

Other meta-heuristic approaches which use EC techniques include ACO [34] and PSO [177, 196]. Sha and Hsu [177] used PSO and Tabu search (HPSO) for JSS problems. HPSO modified the particle position based on priority-based and a new preference list-based representation. The experimental results showed that HPSO has better solutions than other meta-heuristics methods.

Xia and Wu [197] developed a hybridization of PSO and simulated annealing. In this work, a hybridization approach is applied to the multi-objective flexible job-shop problems with makespan, total workload, and critical machine workload. The results showed that the hybridized approach effectively solved problems of up to 20 machines and 20 jobs.

Hyper-heuristics

As described by Cowling et al.[37] as "heuristics to design heuristics", hyper-heuristics have gained the attention of researchers whose goals are to design generic but effective heuristic methods to problems [20].

Burke et al. [20] describe one of the main motivations for developing hyper-heuristic approaches to handle the "challenge of automating the design and tuning of heuristic methods to solve hard computational search problems." Hyper-heuristics are designed to work on low-level heuristic components with the help of suitable operators that combine low-level heuristics and a method of evaluating the performance of heuristics. The

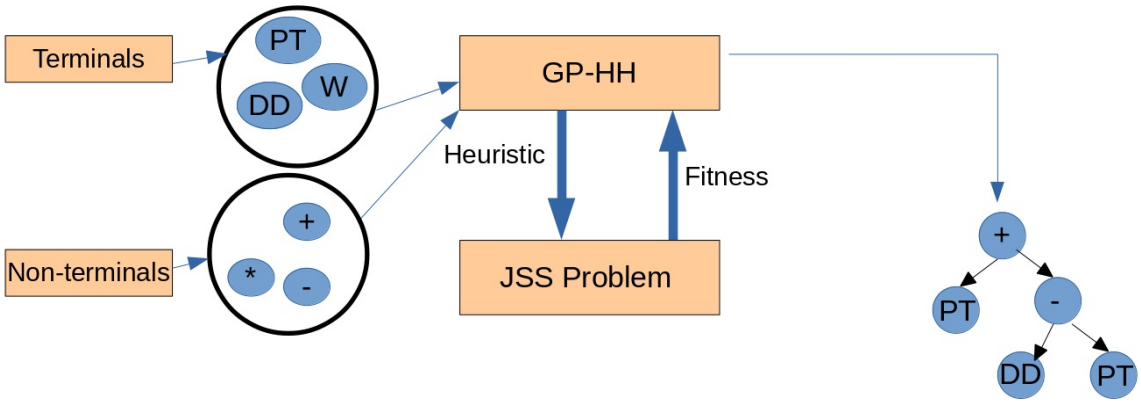


Figure 2.6: Overview of a tree-based GP-HH applied to JSS.

hyper-heuristic methods hence search in the heuristic space [21]. This enables us to easily distinguish hyper-heuristics from meta-heuristics as meta-heuristics search in the solution space.

One popular hyper-heuristic approach is GP-HH [20]. Figure 2.6 shows a high-level overview of a tree-based GP being applied to a JSS problem. Figure 2.6 shows a tree-based GP-HH takes in the base heuristics $\{PT$ (processing time), W (weight), and DD (due date)$\}$ and the operators $\{+, -, \times\}$ to initialize the individuals in the population. The GP system then passes the heuristic into the JSS problem domain and gets a goodness measure back in the individual's fitness. The final output is a heuristic which represents a priority-based dispatching rule $PT + DD - PT$.

2.2.2 GP-HH for JSS

The main aim of hyper-heuristic is to automate designing and selecting the heuristics to solve hard and complex problems such as JSS problems [20]. GP is able to evolve complex programs or rules. It becomes a suitable paradigm for learning heuristics known as GP-HH [20].

Representation of GP as a Hyper-Heuristic

In GP, the most prominent representation and traditionally represented as tree structures. Other program structures have been investigated in the literature, such as linear sequences of instructions or grammar.

The individual of tree-based GP is represented as arithmetic function trees [105, 145, 159] which are called priority dispatching rules in the scheduling environment. Hunt et al. [74] used arithmetic representation to evolve dispatching rules to minimize tardiness objectives in the static JSS problem. The evolved rules from GP were more effective than manual design dispatching rules.

Nguyen et al. [145] investigated three GP representations for JSS. The first representation (R_1) evolved decision trees, the second representation (R_2) used arithmetic representation, and the third representation (R_3) showed a mixed representation, which combined (R_1 and R_2). Experimental results showed that the evolved rules outperformed the existing rules on the static JSS problems [145].

GP-based hyper-heuristics for scheduling

The first comprehensive review on hyper-heuristics is shown in [17]. This study focused on the key design choices and critical issues involved in the process of developing scheduling heuristics. Hunt et al. [75] proposed a GP-HH approach to evolve dispatching rules for both the static and dynamic two-machine job shop environments. This was considered the first study that used GP to evolve dispatching rules for reducing makespan.

Park et al. [158] improved the robustness of evolving ensembles of dispatching rules from a single population of GP individuals. They investigate four ensemble approaches based on majority voting, linear combination, weighted linear combination and weighted majority voting. Through experimentation, they found that linear combination schemes are better than other ensemble techniques. Hunt et al. [73] revealed that local infor-

mation is one of the drawbacks in dispatching rules. Therefore, they used lookahead terminals for GP to evolve "less myopic"rules for dynamic job shop scheduling problems.

Nguyen et al. [145] proposed a tree-based GP representation for evolving rules for static JSS problems. They proposed different representations of dispatching rules and suggested that the representation that integrates the system and machine attributes can improve the quality of the evolved rules. Nguyen et al.[141] also developed methods for evolving iterative dispatching rules using GP.

Tay and Ho [185] used GP-HH for multi-objective FJSS problems. They tried to minimize makespan, mean tardiness, and mean flowtime. They used an aggregation method to solve the multi-objective FJSS problems. Nguyen et al.[147] developed a GP-HH method for multi-objective JSS problems that optimize five conflicting objectives simultaneously.

A literature survey shows that dispatching rules are prominently used for solving JSS problems with a single objective or multiple objectives. There are no many works in the literature to deal with many-objective JSS problems. However, the previous section mentioned that Pareto-based EMO algorithms may not effectively handle many objectives due to a high proportion of non-dominated solutions. Therefore, in this research, we will use GP-HH with the selection scheme of NSGA-III for evolving the dispatching rules more effectively.

2.2.3 Multi-objective and many-objective JSS

In the literature, many research works have been used EC algorithms for JSS but these algorithms mainly focus on optimizing a single objective [67, 159]. For example, GAs have been utilized for minimizing makespan while scheduling identical parallel machines [99]. Park et al. have also considered using cooperative evolutionary technologies to minimize the TWT in dynamic JSS problems [159].

It is becoming clear that JSS essentially has multiple (or many) different objectives. In general, there are two alternative approaches for handling multiple objectives in a job shop, i.e., the *aggregation method* and the *Pareto-dominance method* [96]. In a typical aggregation method, multiple optimization objectives have to be aggregated together to form a scalar-valued fitness function through a weighted sum [116]. Obviously, this method's usefulness is restricted to the situation when the preferences over different objectives can be quantified before applying any EC techniques. On the other hand, without using any aggregation functions, the *Pareto- dominance* concept can be exploited to define the optimization criteria for a guided search of *Pareto-optimal schedules* [116]. Based on this idea, many EC algorithms have been proposed with the aim of evolving the *Pareto-front* [43, 44, 211, 27, 76].

In the literature, the Pareto-dominance method has attracted substantial research attention. Prominent examples include the NSGA-II and SPEA2. Specific techniques have also been successfully developed for multi-objective JSS. For instance, Murata et al. [137] proposed a multi-objective GA for flowshop scheduling problems. Their research specifically considered problem instances with concave Pareto-front and at most three optimization objectives (i.e., the makespan, total tardiness, and total flowtime). A trade-off between the makes and the availability of machines identifies in bi-objective JSS problems [200]. Nguyen et al. [147, 144] used several multi-objective GP approaches in order to evolve scheduling rules consisting of a dispatching rule and due-date assignment rule. Those approaches evolved two expression trees, one of which would be used for due-date assignment to jobs, while the other would be used as a standard dispatching rule. The results showed that the evolved rules outperformed various combinations of existing scheduling rules from the literature. Evolving dispatching rules for multi-objective criteria were also analyzed by Nguyen et al. [146]. In [73], a scheduling problem consisting of two and three scheduling criteria was optimized by the use of two pro-

posed heuristics and a genetic algorithm.

Moreover, A short overview of some other multi-objective problems in the unrelated machine's environment can be found in [163]. Moslehi and Mahnam [128] present an approach based on a hybridization of the particle swarm and local search algorithm to solve the multi-objective flexible JSS problem. Zhang et al. [207] have formulated the textile dyeing process scheduling problem as a bi-objective optimization model, in which one objective is related to tardiness cost while the other objective reflects the level of pollutant emission. And they proposed a multi-objective particle swarm optimization algorithm enhanced by problem-specific local search techniques (MO-PSO-L) to seek high-quality non-dominated solutions. Luo et al. [126] proposed a distributed flexible JSS Problem with transfers (DFJSPT), in which operations of a job can be processed in different factories. In order to expand the search space and accelerate the convergence speed of the solution, an efficient memetic aAlgorithm (EMA) is proposed to solve the DFJSPT with the objectives of minimizing the makespan, maximum workload, and total energy consumption of factories. Zhao et al. [209] proposed an improved multi-objective evolutionary algorithm, which is based on decomposition (IMOEA/D) for multi-objective JSS problem which minimized three objectives – the maximum completion time (makespan), the total flow time and the tardiness time are considered simultaneously. Several prior rules are presented in the proposed algorithm to construct the initial population with a high-quality level. Karunakaran et al. [98] proposed the sampling heuristic for GP-HH and presented the algorithm based on the island model approach.

In addition to the research works mentioned above, some researchers have started considering JSS problems with more than three objectives. For example, Nguyen et al. [147] have considered designing dispatching rules for general JSS problems with up to five different objectives. In [137], four different objectives have been considered for evolving optimal schedules. Fowler et al. [51] considered the problem for scheduling

a printed wiring board manufacturer's drilling operation subject to five optimization objectives. They have used several approaches to solve the given problem. Kolahan and Kayvanfar [103] used the simulated annealing approach to solve a scheduling problem consisting of the makespan, earliness, and tardiness objectives. Wang et al. [192] proposed an algorithm for many-objective flexible job shop scheduling problems. In this algorithm, the mathematical model was established considering six objectives: makespan, workload balance, mean of earliness and tardiness, cost, quality, and energy consumption simultaneously. XU et al. [198] proposed an algorithm for many-objective flowshop scheduling problems based on fuzzy sets' relative entropy. This algorithm built a mathematical model of the many-objective flow-shop scheduling problems, including four objectives, makespan, tardiness, total inventory cost and total tardiness cost. A many-objective permutation flow shop scheduling problems model with four objectives, namely, the makespan, total tardiness, inventory holding cost, and energy consumption cost, was established in [210]. Gong et al. [56] considered many-objective flexible job shop scheduling problem (MaOFJSP) with five objectives under dynamic electricity pricing and applied non-dominated genetic algorithm-III to solve it.

Our survey found that there has been relatively little attention to solving many-objective JSS problems. There are a few studies found in the literature of many-objective JSS but these studies used conventional MOEAs. MaOPs pose a number of challenges to Pareto-based MOPs [43, 31]. First, the proportion of non-dominated solutions in a population rises rapidly with the number of objectives. This issue also becomes a part of JSS problems whenever optimization of more than three objectives is attempted. Second, a commonly used diversity-preservation operator such as crowding distance [44], clustering [211] cannot strengthen the Pareto front's selection pressure. These operators also become computationally expensive [84].

2.2.4 Local search for JSS

Local search is a popular method for dealing with NP-hard problems — the first application of a local search that was studied in [38]. There are several reasons for interest in local search algorithms. One important aspect is that local search algorithms are more easily understandable and implementable than exact algorithms. In practice, local search has been shown to be very effective at solving complicated problems [1].

The local search algorithm is widely used for solving JSS [166]. It was first introduced for JSS in 1980. A year later, Talliard et al. [183] introduced the tabu search for JSS problems. One of the studies [189] discussed the local search method to emphasize deterministic and randomized JSS problems. They also discussed existing neighborhood techniques for local search methods. Hunt et al. [75] applied a local search in the fitness evaluation process to improve the performance of dispatching rules. Furthermore, JSS problems with alternative sequence and sequence-dependent setup are addressed in [33]. They developed mixed-integer program models with local search to reduce computational time. The research study in [58] solved stochastic JSS problems using several neighborhood functions from the literature.

A few pieces of literature show that local search is also used with GP to improve the quality of evolved dispatching rules [148, 173]. Nguyen et al. [148] improved the quality of evolved dispatching rules for dynamic JSS through Iterated Local Search (ILS) [124]. The key idea of ILS is to iteratively apply local search heuristics to solutions obtained by perturbations of previously visited locally optimal solutions [10]. The study in [173] combined ILS, and GP. The goal of GP in these algorithms was to evolve perturbation operators based on a range of low-level operators and rules.

2.2.5 PLS for JSS

PLS is one of the heuristic algorithms for handling combinatorial optimization problems and maintaining the Pareto archive [155]. The goal of

PLS is to approximate a high-quality Pareto-optimal set. It can be used as a standalone algorithm [156] or hybridized with the EMO algorithm [127].

Hybridization of local search with EMO is often called a memetic algorithm [81]. This hybridization improves the searchability of the pure EMO algorithms [88]. The hybrid algorithms have been found in the literature for solving combinatorial optimization problems effectively [14, 49, 82, 88, 81, 137]. The evaluation mechanisms in the local search of memetic EMO algorithms are performed by either weighted scalar fitness function [81] or Pareto ranking [102]. The effectiveness of PLS as a component of the hybrid algorithms may be explained by the fact that it has different characteristics than most other multi-objective algorithms. As noticed by Lara et al. [108], each multi-objective metaheuristic should search both towards and along the Pareto-front.

Several methods of PLS have been proposed in the multi-objective JSS problems. Murata et al. [137] proposed a multi-objective GA to minimize the makespan, total tardiness, and flowtime. Ishibuchi et al. [81] proposed a genetic local search algorithm. This algorithm uses a weighted sum of multiple objectives as a fitness function to guide parents' selection for generating offspring solutions through crossover and mutation operations. A local search is then employed to each solution generated by genetic operators to improve its fitness value further. This algorithm was tested on two instances of flowshop with two pairs of objectives (makespan and total tardiness) and (makespan and maximum tardiness). Moslehi et al. [136] proposed a hybridization of a particle swarm algorithm and local search methods for multi-objective JSS problems. In this algorithm, local search is employed to enhance convergence speed. The research study in [4] defined a variable neighborhood search (VNS) for a multi-objective dynamic JSS problem. In this approach, VNS is hybridized with an artificial neural network (ANN), where ANN is used to optimize the algorithmic parameters for VNS. They showed that VNS can outperform common human-made dispatching rules such as SPT and FIFO.

The research in [89] applied PLS to the many-objective combinatorial optimization problems and showed high effectiveness of the proposed many-objective Pareto local search algorithm. In this algorithm, they used three new mechanisms: (1) the efficient update of large Pareto archives with ND-Tree data structure, (2) a new mechanism for the selection of the promising solutions for the neighborhood exploration, and (3) partial exploration of the neighborhoods. Furthermore, Seada et al. [175] combined NSGA-III with local search and Karush-Kuhn-Tucker Proximity Measure (KKTPM). The local search in this algorithm used two distinct operators. One operator enhances overall population diversity and the other promotes convergence.Meanwhile, KKTPM speeds up the convergence and guides the second local search operator.

2.3 Summary

This chapter has discussed the key concepts of GP, JSS, hyper-heuristics, multi-objective optimization, many-objective optimization, model-based EMO, Gaussian process, local search, and PLS. GP is one of the most popular hyper-heuristic because of its flexibility, which automatically evolves dispatching rules in single and multi-objective JSS. However, the research in the direction of many-objective JSS is relatively new and many aspects need to be explored to enhance the quality of GP-HH methods and cope with practical requirements. There are some identified limitations of the existing body of work in the literature, which motivate this thesis's work. These limitations are:

- There are a number of different approaches that have been proposed in the literature for various JSS problems. Most of the works in the literature on JSS do not focus on many-objective JSS. Our survey found that there has been relatively little attention to solving many-objective JSS problems. Heuristic approaches are prominently used

to solve JSS problems because of its advantages over other methods. Literature shows that heuristic approaches are better for solving multi-objective JSS problems. Moreover, GP-HH approaches have been successfully used to design effective dispatching rules automatically. However, GP-HH approaches are relatively new for many-objective JSS and require further investigations. There are a few studies found in the literature of many-objective JSS but these studies used conventional MOEAs. These MOEAs fail to work properly for MaOPs. Due to the issue of using MOEAs in solving many-objective JSS, new GP-HH approaches are required that can potentially work with many-objective JSS.

- Literature survey shows that decomposition-based approaches are good for diversity maintenance [43]. The decomposition-based approaches use uniformly distributed reference points and perform better on problems which have uniformly distributed Pareto-front such as the DTLZ1 problem. However, they face difficulty when it is applied to non-uniform and irregular Pareto-front problems such as the DTLZ7 and JSS problems. Deb and Jain [86] also notice this limitation. Deb and Jain have witnessed several many-objective problems where reference points can never be associated with any good solutions, while others are associated with more than one candidate solutions. The distribution of the Pareto-front of JSS problems in the objective space is also non-uniform. Algorithms for many-objective JSS are required to consider this limitation and generate reference points according to the distribution of the Pareto-front.

- Literature survey shows that local search is a widely used technique for the classical combinatorial optimization problem. There are several existing studies using PLS (extension of the local search) for multi-objective JSS problems. These algorithms hybridized the local search with the global search and used to improve the quality of

the dispatching rules. This hybridization is called the memetic algorithm. PLS enhances multi-objective GA performance; however, no works in the literature hybridize GP with PLS techniques. In this thesis, we enhance the quality of evolved dispatching rules through PLS in many-objective JSS.

The following contribution chapters in this thesis will propose new approaches that mainly aim to address these limitations.

Chapter 3

Experimental Methodology

This chapter presents the experimental methodology of this thesis. First, it discusses the benchmark problems for JSS that will be used throughout the thesis. Next, it describes the GP functions and terminals for JSS. Then it discusses the performance measures used for evaluating many-objective optimization algorithms.

3.1 Benchmark problems for JSS

There are four widely used static job shop benchmark sets in the literature. These are Lawrence static job shop benchmark set (LA 01-40 Lawrence) [113], Applegate and Cook benchmark set (ORB 01-10) [8], Taillard static job shop benchmark set (TA 01-80) [182], and Demirkol et al. benchmark set (DMU 01-80) [46]. Details of the benchmark sets can be seen in Table 3.1.

Table 3.1: Static JSS data sets.

Data set	Notation	# of instances	Size (#J_#M)
LA	LA01-LA40	40	10 _ 5 to 15_ 15
ORB	ORB01-ORB10	10	10_ 10
TA	TA01-TA80	80	15_ 15 to 100 _ 20
DMU	DMU01-DMU80	80	20 _ 15 to 50 _ 20

The goal of this thesis is to evolve effective dispatching rules (priority functions) for many-objective JSS. We used static JSS environments because many-objective research is comparatively newer than single objective JSS and multi-objective JSS. Further, dynamic JSS environments are more complex than static JSS environments.

The TA is the most used benchmark problem for solving static JSS problems in the literature [131, 159]. The benchmark targets permutation flowshop, flowshop, open shop, and JSS problems. It contains job shop instances of sizes up to 100 jobs on 20 machines, resulting in 2000 job operations. This benchmark problem is still using to use to solve multi-objective optimization problems [97]. Therefore, the TA has been selected as the dataset for experiments in this thesis.

There are 80 indexed problem instances in the TA set (i.e., each problem instance has a different ID from 1 to 80). These can be further divided into eight groups (denoted as TA-1, TA-1, ..., TA-8). The problem instances in the same group have the same number of jobs and machines but the processing time matrices are generated by using different random seeds. In the experiments, the total 80 instances are divided into the *training set* and the *test set*, each consisting of 40 instances. The training set consists of all the instances with odd IDs and the test set contains all the instances with even IDs.

The GP uses the training instances in order to evolve dispatching rules for some given objectives. Once obtained, the effectiveness of these evolved dispatching rules will be evaluated by using the test instances. The release times of all jobs in all problem instances are safely set to zero. Since there is no due date information included in the original dataset, the due date assignment rule [11] will assign the due dates of each insistence of jobs. That is,

$$dd(j_i) = \lambda \times \sum_{k=1}^{m} p_i^k.$$

where $dd(j_i)$ stands for the due date of the job j_i, and λ is used to indicate

the tightness of due dates. We chose λ value is 1.3 for all instances in the training set and test set in our experiments because it is most commonly used value in the JSS literature. The job weights are set according to the 2:6:2 rule [165]. That is, the weight is set to 4, 2, and 1 with the probabilities of 20%, 60%, and 20%, respectively. These weighted probabilities in this research are inspired by Pinedo and Singer [165]. Their study showed that the first 20% of jobs are assigned a weight of 4, the weight of 2 is assigned to the next 60% of jobs, and weight of 1 is assigned the last 20% of jobs.

Four objectives are considered in the experiment. These are the mean flowtime (mF) (see equation (2.1) of Chapter 2) , maximal flowtime (maxF) (see equation (2.2) of Chapter 2), mean weighted tardiness (mWT) (see equation (2.6) of Chapter 2) and maximal weighted tardiness (maxWT) (see equation (2.7) of Chapter 2).

3.2 GP terminals and function set

The terminal set is described in Table 3.2. The non-terminal set is set to $\{+, -, *, /, min, max, ifthe\}$, where "+", "−" and "$*$" are basic arithmetic operators. The function "/" is the protected division operator which returns one if the denominator is zero. The functions "min" and "max" take two arguments and return either the smallest and the largest value, respectively. The function $ifthe$ takes three arguments a, b and c, and returns b if $a > 0$ and c otherwise. These non-terminals [141] used by existing GP-HH methods in the literature and also used in manually designed rules [20, 70]. These rules determine the priority of jobs by computing the expression. For example, the simple shortest processing time (SPT) rule [176] and the apparent tardiness cost (ATC) rule [70].

3.3 Performance measures

There are a variety of performance measures proposed for evaluating multi-objective optimization algorithms from different perspectives. In

Table 3.2: Terminal set of GP for JSS.

Attribute	Symbol
Job Properties	
Processing time of the operation	PT
Processing time of the next operation	NOPT
Ready time of the next machine	NMRT
Work Remaining	WKR
Work in the next queue	WINQ
Number of operations in the next queue	NOINQ
Flow due date	FDD
Due Date	DD
Weight	W
Machine Properties	
Number of operations in the queue	NOIQ
Work in the queue	WIQ
Ready time of the machine	MRT

this thesis, we choose the *HyperVolume* [212] and *Inverted Generational Distance* [206]. They are the two main performance measures in the literature. Theoretically, a set of trade-off dispatching rules with better performance should have a *larger* HV value and a *smaller* IGD value. All the objectives have been normalized into the range between 0 and 1 before calculating the above performance measures.

HyperVolume (HV)

Given a nadir points $\boldsymbol{z}^*$ and a set of non-dominated solutions $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n\}$ in the objective space. HV indicates the area covered by $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n\}$ with respect to $\boldsymbol{z}^*$.

The selection of the nadir points is an important issue. It is better for $\boldsymbol{z}^*$ to be slightly larger than the maximum value of each objective to emphasize the balance between convergence and diversity. In the thesis, all algorithms have normalized objective values. So that the ideal point and the nadir point are (0, 0) and (1, 1) respectively in the objective space of a two-objective problem. After this normalization, the nadir points ($\boldsymbol{z}^*$, $\boldsymbol{z}^*$) with $\boldsymbol{z}^*$ =1 + $\frac{1}{(n-1)}$ (where n is the size of solutions) may be a good choice

when the Pareto front is nonlinear or irregular [78].

For HV, the nadir point is set to (1, 1, 1, 1) in this thesis since all the objectives are to be minimized.

Inverted generational distance (IGD)

The IGD value is based on the true Pareto-front $\{\boldsymbol{y}_1^*, \ldots, \boldsymbol{y}_u^*\}$. Specifically, it is defined as the average distance of each Pareto-optimal solution $\boldsymbol{y}_i^*$ to its closest point in $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_s\}$. That is,

$$IGD = \frac{1}{u} \sum_{i=1}^{u} \min_{\boldsymbol{y} \in \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_s\}} \{dist(\boldsymbol{y}_i^*, \boldsymbol{y})\}. \tag{3.1}$$

Since the true Pareto-front is unknown in JSS, an approximate Pareto-front will be adopted for IGD by selecting the non-dominated solutions among the final solutions acquired from all the runs of all the tested algorithms.

Chapter 4

Many-Objective GP-HH for JSS

4.1 Introduction

A study of the literature shows that most of the past research on JSS focuses on optimizing job shop scheduling (JSS)-relevant objectives [67]. For example, Park et al. have considered cooperative evolutionary technologies to minimize the total weighted tardiness (TWT) in dynamic JSS problems [159]. It is technically simple to consider only a single optimization objective for JSS. However, it is now widely evidenced in the literature that JSS by nature presents conflicting objectives, including the makespan, mean flowtime, maximum tardiness, maximum lateness, total workload, and proportion of tardy jobs [137, 122].

Over the last few years, JSS has been frequently considered as a multi-objective optimization problem with several potentially conflicting objectives [201]. The research study in [185] presented the first work on genetic programming (GP) that focuses on multi-objective JSS problems. In this study, an aggregation method was used which aggregated the multiple optimization objectives together and formed a scalar-valued fitness function through a weighted sum [79]. However, this method is restricted to the situation when the preferences over different objectives can be quantified before applying any evolutionary computation (EC) techniques. The

Pareto-dominance approach can be exploited to define the optimization criteria for a guided search of Pareto-optimal schedules. In the literature, the Pareto-dominance approach has attracted substantial research attention. Prominent examples include the non-dominated sorting genetic algorithm II (NSGA-II) [44] and the strength Pareto evolutionary algorithm 2 (SPEA2) [211].

Despite the rapid development of EC techniques for multi-objective JSS, existing studies of JSS problems that have more than three objectives (so-called *many-objective*) are still very limited. There are only a few studies on many-objective JSS [53, 147]. In [147], Nguyen et al. considered five conflicting objectives to design dispatching rules for JSS problems. These studies used multi-objective evolutionary algorithms (MOEAs) such as NSGA-II [44] and SPEA2 [211] for solving the many-objective JSS Problems. However, MOEAs noticeably deteriorate their search ability when more than three objectives are involved [31, 43]. One major reason for this is that the proportion of non-dominated solutions in a population rises rapidly with the number of objectives. This issue also becomes a part of JSS problems, whenever optimization of more than three objectives is attempted. Therefore, we aim to explicitly investigate JSS problems with many conflicting objectives and consider the challenges of many-objective optimization.

As we discussed in Chapter 1, rather than directly evolving Pareto-optimal schedules, we evolve dispatching rules. We hope that the evolved dispatching rules could achieve a good balance over many conflicting objectives. This is essentially a machine learning approach where rules can be evolved based on a training set of problem instances and then further evaluated on a different set of testing instances. In the literature, the most effective technique is to evolve dispatching rules using (GP-HH) [143, 148] which is also be adopted in this study. This chapter aims to use *GP-HH for evolving dispatching rules for many-objective JSS problems, which can provide potential trade-offs among different objectives.* The sub-objectives of this chap-

ter are:

1. To develop a many-objective GP-HH method to evolve dispatching rules for JSS,

2. To investigate the relationship among JSS objectives,

3. To perform detailed comparisons with other multi-objective evolutionary algorithms (i.e. NSGA-II [44] and SPEA2 [211]) for verifying the efficacy of the proposed algorithm in solving many-objective JSS.

The first sub-objective provides the algorithm for evolving a set of trade-off dispatching rules in many-objective JSS. A few interesting many-objective algorithms have already been developed in recent years [116, 43]. Among all the different technologies, we have particular interests in the decomposition-based approach (see in Section 2.1.5 of Chapter 2). NSGA-III is a state-of-the-art algorithm that can be seen as a decomposition-based approach and has shown leading performance on many benchmark problems [43]. In this thesis, we used decomposition-based approaches specifically for obtaining Pareto-optimal solutions for many-objective JSS problems. The second sub-objective will give an answer to the question of how different scheduling objectives influence each other when they are optimized together. Finally, the last sub-objective will examine multi-objective optimization algorithms such as NSGA-II and SPEA2 are competent at tackling many-objective JSS problems. The first research objective's main goal is to use GP-HH for JSS problems to find the Pareto fronts of non-dominated dispatching rules to deal with many conflicting objectives. NSGA-II and SPEA2 were selected since they represent one of the most popular Pareto dominance based multi-objective algorithm. Also, the basic framework of the NSGA-III remains similar to the original NSGA-II algorithm. Further, NSGA-II and SPEA2 used in [147] and we will explore the performance of the proposed many-objective algorithm as compared to the NSGA-II and SPEA2 algorithms in solving many-objective JSS. In

Section 4.2, a problem description is presented. Section 4.3 describes the proposed algorithm for many-objective JSS. Section 4.4 gives a description of the experimental studies. Section 4.5 describes the achieved results and contains the discussion. Finally, Section 4.6 concludes this chapter.

4.2 Problem description

Based on the JSS problem described in Subsection 2.1.2 of Chapter 2, our goal is to evolve useful dispatching rules that can build the complete schedules incrementally. The quality of the schedules created by using these rules will then be evaluated with respect to D objectives $\boldsymbol{f} = \{(f_1, f_2, \ldots, f_d)\}$.

Without losing generality, we assume that all the dimensions of $\boldsymbol{f}$ are minimized. Here, we consider the case with $D \geq 4$, i.e., there are *four or more* objectives (*many-objective* JSS). Given two schedules Δ_1 and Δ_2, it is said that Δ_1 *dominates* Δ_2 if and only if

$$\forall i, 1 \leq i \leq D, f_i(\Delta_1) \leq f_i(\Delta_2) \tag{4.1}$$

and

$$\exists i, f_i(\Delta_1) < f_i(\Delta_2). \tag{4.2}$$

Consequently, if P_1 is the dispatching rule that produces schedule Δ_1 and P_2 is the dispatching rule that produces schedule Δ_2, then we can also say that, for the given problem instance I, P_1 dominates P_2. Based on the above explanations, it should now be clear that our goal is to evolve a collection of *Pareto-optimal* dispatching rules, jointly known as the *Pareto-front*. Each dispatching rule in the Pareto-front is non-dominated by any other rules evolved by using our proposed algorithm.

In the job shop, when a new job arrives at an idle machine that job will be processed immediately. The information about that job will be extracted from the terminals in Table 3.2 of Chapter 3. Then, the dispatching rule will be evaluated and after the evaluation, priority will be assigned

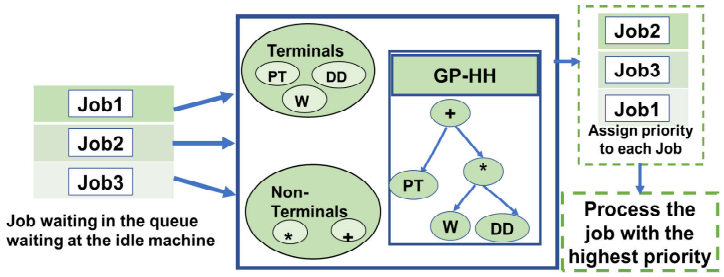


Figure 4.1: Illustration of a dispatching rule in JSS.

to the considered job. This procedure will be applied until priorities are assigned to all waiting jobs on the machine and the job in the queue with the highest priority will be processed next. This can be seen in Figure 4.1.

4.3 Many-objective-GP-HH for JSS

This section shows how the proposed Many-Objective-GP-HH method is used to solve many-objective JSS problems. It is common to use hyper-heuristics for evolving dispatching rules for JSS and this has achieved great success [68, 87]. Section 4.3.1 will show how GP represents dispatching rules. The newly proposed algorithm which is named GP-NSGA-III is given in Algorithm 3. GP-NSGA-III combines the initialization, evaluation, and evolutionary operators of GP and the selection scheme of NSGA-III. Then, the proposed Many-Objective-GP-HH algorithm is presented.

4.3.1 Representation of rules

As with previous studies of GP for JSS problems [143, 147], dispatching rules are also represented by GP trees [105] in this thesis. Basically, dispatching determines the priorities of jobs waiting in the queue. Figure 4.2 shows the GP tree representation of the 2PT+WINQ+NPT rule [70].

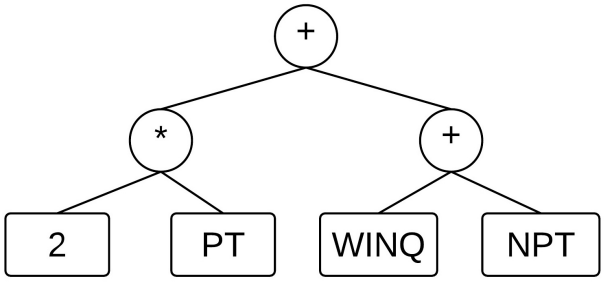


Figure 4.2: The GP tree representation of the 2PT+WINQ+NPT rule.

In Figure 4.2, the terminals in that tree are {2, PT, WINQ, NPT}, and the non-terminals are $\{+, *\}$. Proper terminal and non-terminal sets are crucial for constructing promising and concise search space for GP. GP-based approaches in this thesis use a mixture of terminal sets. These terminals are used by existing GP-HH approaches in the literature and evolved quality rules [141, 143, 147]. These terminals range is from common attributes (e.g., operation processing time PT) to more complex terminals (e.g., processing time of the next operation NPT). WINQ and NPT reflect the status of the current machines. Whereas WINQ denotes the sum of imminent process times of all jobs waiting at the queue of the next operation of job i. The commonly considered job shop attributes are included in this thesis, which has been summarized in Table 3.2 of Chapter 3.

4.3.2 General framework of GP-NSGA-III

In this thesis, we evolve dispatching rules to minimize four popular JSS objectives which include the mean flowtime (see equation (2.1) of Chapter

Algorithm 3: The framework of GP-NSGA-III.

Input : A training set $\boldsymbol{I_{train}}$
Output: A set of non-dominated rules $\boldsymbol{P}^*$
Initialize and evaluate the population $\boldsymbol{P}_0$ of rules by the ramped-half-and-half method;
Calculate set of reference points $\boldsymbol{Z}$;
Set $P_g \leftarrow P_0$ and $generation \leftarrow 0$;
while $generation < maxGeneration$ **do**
    Generate the offspring population Q_g using the crossover, mutation and reproduction of GP;
    foreach $Q \in Q_g$ **do**
        Evaluate rule Q
    end
    $\boldsymbol{R}_g \leftarrow \boldsymbol{P}_g \cup \boldsymbol{Q}_g$;
    $(F_1, F_2, \cdots) \leftarrow$ Non-dominated-sort(R_g);
    Form the new population $\boldsymbol{P}_{g+1}$ from $(F_1, F_2, \cdots)$ by the NSGA-III selection;
    $generation \leftarrow generation + 1$;
end
return *The non-dominated individuals* $P^* \subseteq P_{g_{\max}}$;

2), maximal flowtime (see equation (2.2) of Chapter 2), mean weighted tardiness (see equation (2.6) of Chapter 2), maximal weighted tardiness (see equation (2.7) of Chapter 2).

The GP-NSGA-III algorithm explores the Pareto-front of non-dominated dispatching rules regarding the four objectives mentioned above. Algorithm 3 shows the workflow of our proposed many-objective algorithm. In this algorithm, first, a number of training instances are given as input. The initial GP population P_0 is created using the ramped-half-and-half method [105]. The ramped half-and-half is the best generative method that works best over a broad range of problems and generates a

wide variety of tree sizes and shapes [105]. All individuals in the GP population P_0 will be evaluated by applying the rules to training instances. This fitness evaluation of GP population P_0 is required in line 1 of Algorithm 3.

Algorithm 4: The evaluation of a dispatching rule.

Input : A training set $\boldsymbol{I_{train}}$ and a rule P
Output: The fitness $\boldsymbol{f}(P)$ of the rule P

1 **foreach** $I \; in \; \boldsymbol{I_{train}}$ **do**
2 Construct a schedule $\Delta(P, I)$ by applying the rule P to the JSS instance I;
3 Calculate the objective values $\boldsymbol{f}(\Delta(P, I))$;
4 **end**
5 $\boldsymbol{f}(P) \leftarrow \frac{1}{|\boldsymbol{I_{train}}|} \sum_{I \in \boldsymbol{I_{train}}} \boldsymbol{f}(\Delta(P, I))$;
6 **return** $\boldsymbol{f}(P)$;

The detailed process for fitness evaluation is presented in Algorithm 4 where $\boldsymbol{f}(.)$ is the objective vector. The quality of each individual (dispatching rule) in the population will be measured by the average value of the objectives across all training instances. This is shown in line 5 of Algorithm 4.

After all, individuals have been evaluated then the reference points Z are generated in Algorithm 3. In this study, we use Das and Dennis's systematic approach [39]. In this approach, reference points to place on a normalized hyperplane (for more detail, refer to Subsection 2.1.6 in Chapter 2). Next, in the algorithm, offspring (Q_g) of generation g is generated by applying subtree crossover and subtree mutation to the current population (P_g). The fitness evaluation of Q_g is required in line 7 of Algorithm 3. The detailed process for fitness evaluation is presented in Algorithm 4. After the fitness evaluation of Q_g, GP-NSGA-III combines the offspring population and the parent $\boldsymbol{P_g}$ together. Later, all the individuals in the combined population R_g are sorted according to different non-domination

ranks (F_1, $F_2 \ldots$, F_{l-1}, F_l). Then, each non-domination rank is selected one at a time to construct a new population of next-generation P_{g+1}, starting from F_1 to F_l or until the size of the population reaches to N (population size). All solutions from $F_{(l+1)}$ to onwards are rejected from the combined population R_g. If F_l cannot be fully inserted into the population, then each individual of F_l is assigned to a reference point $Z \in \boldsymbol{Z}$ and selected one by one using the niching method (details of the niching method can be seen in Subsection 2.1.5 of Chapter 2) and in [43].

4.4 Experimental studies

In this section, experimental studies will be conducted to compare the proposed GP-NSGA-III to GP-NSGA-II and GP-SPEA2 which are two well-known MOEAs. All compared algorithms are combined with GP as well, i.e., they adopted the solution representation and evolutionary operators of GP and selection mechanisms of NSGA-II and SPEA2 (The detailed information of NSGA-II and SPEA2 can be seen in Section 2.1.4 of Chapter 2). In the following, we will describe the parameter settings of the algorithms. For each run of each compared algorithm, the experiment consists of two steps as follows.

1. In the first step, the algorithm (GP-NSGA-III, GP-NSGA-II, or GP-SPEA2) is applied to the training set and obtain a set of trade-offs dispatching rules;

2. In the second step, for every problem instance in the test set, we apply every newly evolved rule to produce a schedule and calculate all the considered objective values with respect to the schedule.

The above steps are repeated for the compared GP-NSGA-III, GP-NSGA-II and GP-SPEA2 algorithms. In terms of objectives, four objectives, i.e. the mean flowtime (mF) (see equation (2.1) of Chapter 2) , maximal flowtime (maxF) (see equation (2.2) of Chapter 2), mean weighted

tardiness (mWT) (see equation (2.6) of Chapter 2) and maximal weighted tardiness (maxWT) (see equation (2.7) of Chapter 2) are considered to minimize in the experiment. We selected the Taillard (TA) static JSS benchmark instances [182] as the testbed. (The detailed information of TA can be found in Section 3.1 of Chapter 3).

4.4.1 Parameter settings

The population size for GP-NSGA-II and GP-SPEA2 is set to 1024. This is a reasonably large population size that ensures enough diversity in the population [135, 145]. The elitist archive of size in GP-SPEA2 is set to 512 by following the ECJ [125] guidelines (we used the ECJ platform for implementing our algorithms). GP-NSGA-II builds the archive separately which maintains the current best individuals. On the other hand, GP-SPEA2 archive consists of the specified proportion of its population.

NSGA-III is suggested that the population size is equal to the number of reference points [43]. In addition, the reference points of NSGA-III depends on the number of objectives. In this study, we chose the population size for GP-NSGA-III that are not significantly different from GP-NSGA-II and GP-SPEA2. The population size chooses for GP-NSGA-III is 1025.

The total number of generations is set to 50, commonly used in the literature [145, 164]. The crossover rate is set to 85% and the mutation rate is set to 10%. The reproduction rate is set to 5%. These rates (reproduction, crossover, and mutation) have been previously used in [145]. The maximum depth of dispatching rules is set to eight. This depth is used to restrict the dispatching rule from becoming too large [145]. However, we can choose this maximum depth to decrease the computational times of the GP system and make the evolved rules easier to analyze. Tournament selection with a tournament size of seven is used to select individuals for genetic operations [105]. In the experiments, the two commonly used measures in multi-objective optimization, i.e., IGD [206] and HV [212] are

used to compare the algorithms (The detailed information can be seen in Section 3.3 of Chapter 3).

4.5 Results and discussions

In the experiment, for each of the three algorithms, 30 independent runs are carried out. Then, the mean and the standard deviation of HV and IGD values are reported. First, we will analyze how the different scheduling objectives are correlated with each other. Then, we will present the training performance results followed by testing performance results.

4.5.1 Relationship among scheduling objectives

This section will analyze how the different scheduling objectives are correlated with each other. In order to visualize the interdependencies between different objectives, we used the aggregated Pareto-front, including the non-dominated evolved dispatching rules extracted from Pareto-front generated by all compared algorithms in 30 independent runs on the training instances. Figure 4.3 shows a scatter plot matrix that contains all the pairwise of four objectives (any two plots in a scatter plot matrix are similar with respect to the diagonal and these two axes are interchanged). This plot helps us to understand the trade-offs between objectives and how optimizing one objective influences optimizing another objective.

Figure 4.3 shows that flowtime objectives (maxF-mF) are conflicting in nature meaning that by optimizing one objective, the other objective becomes worse. Therefore, maxF compromises its objective values because of the mF and vice versa. This is because trying to minimize the maxF could negatively affect several other jobs in the shop and thus increase the total mF of the entire system. Also, if we are trying to minimize the maxWT it may increase the mWT of other jobs which are late in the entire system.

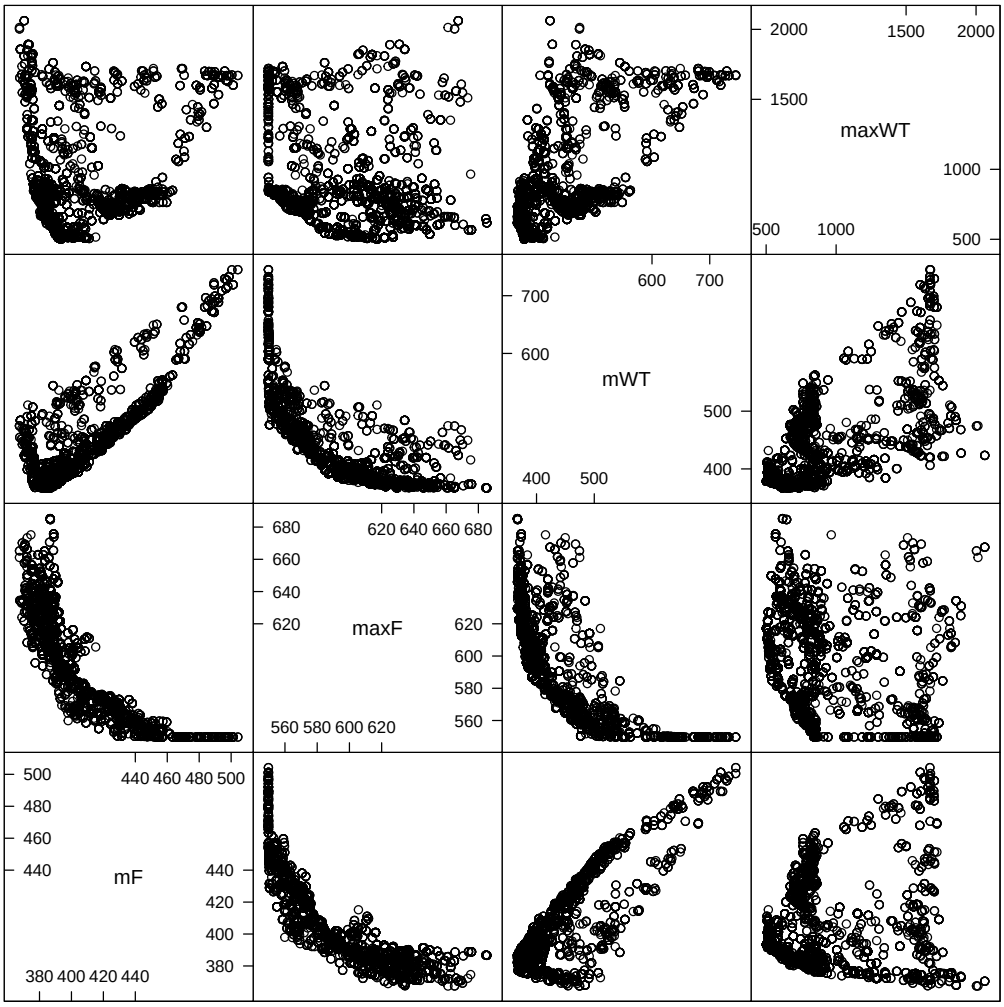


Figure 4.3: Pareto-front for pairwise objective combination.

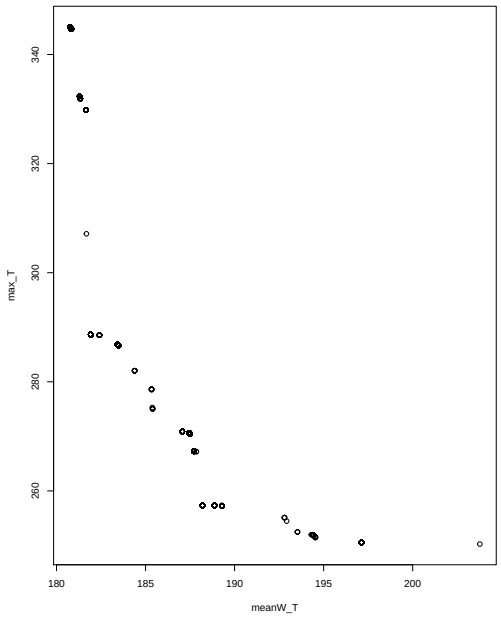


Figure 4.4: Pareto-front for pairwise objectives combination between $(mWT - maxWT)$.

It is interesting to note from Figure 4.3 that mF and mWT have a strong relationship when the values are high. Thus, by optimizing one objective, the other objective will also be indirectly optimized. However, trade-offs between mF and mWT can be witnessed when these objectives reach their lowest extreme. We can see from the Pareto-front that reducing maxWT will deteriorate other objectives. On the other hand, maxF optimized well with mWT and mF.

It can also be observed from Figure 4.4 that the due date related objectives (maxWT-mWT) have overlapping solutions. This could be possible because the jobs which contribute late to the objective value are mostly redundant.

Table 4.1: The mean and standard deviation over the average HV and IGD values on training instances of the compared algorithms in the four-objective experiment.

HV ($\bar{x} \pm \sigma$)		
GP-NSGA-III	GP-NSGA-II	GP-SPEA2
0.694(0.0152)	0.688(0.0220)	0.564(0.0233)
IGD ($\bar{x} \pm \sigma$)		
GP-NSGA-III	GP-NSGA-II	GP-SPEA2
0.00139(0.00021)	0.00163(0.00034)	0.00235(0.00019)

4.5.2 Results

Table 4.1 shows the mean and standard deviations of the training performance in terms of HV and IGD of the rules obtained by GP-NSGA-III, GP-NSGA-II, and GP-SPEA2. For each instance, Wilcoxon's rank-sum test [195] is conducted to compare the best performing algorithm against the other two algorithms. The significance level is set to 0.05. If both p-values are smaller than 0.05, then the best algorithm is considered significantly better than the other two algorithms. The significantly better results are marked in bold.

Performance of Obtained Dispatching Rules

Table 4.1 reveals that GP-NSGA-III outperforms GP-NSGA-II and GP-SPEA2 in terms of both HV and IGD on training instances. However, GP-NSGA-II is very competitive with GP-NSGA-III in terms of HV. For more detail, Tables 4.2 and 4.3 show the test performance on the 40 test instances.

Tables 4.2 and 4.3 show the mean and standard deviation over the HV and IGD values of the 30 independent runs of the compared algorithms with respect to the 40 test instances in the 4-objective experiment. In the tables, "ID" is the instance ID, and #J and #M indicate the number of jobs

and machines, respectively. One can see that as the instance ID increases, the problem size increases as well.

Tables 4.2 and 4.3 together suggest that GP-NSGA-III performed significantly better in terms of both HV and IGD than GP-NSGA-II and GP-SPEA2 in most of the tested instances with four objectives have significantly deteriorated each other. In terms of HV, GP-NSGA-III performed the best with statistical significance in 31 out of the total 40 instances. GP-NSGA-II outperformed the other algorithms in the remaining 8 instances. GP-SPEA2 failed to achieve the best performance in any instance.

In terms of IGD, Table 4.3 reveals a similar pattern. GP-NSGA-III performed significantly better in 37 out of the 40 instances, and GP-NSGA-II achieved the best performance in the remaining 3 instances. Again, there is no instance in which GP-SPEA2 performed the best.

When taking a closer look at the tables, it can be found that GP-NSGA-II performed better on smaller instances (no more than 20 jobs and 20 machines). On the other hand, GP-NSGA-III appeared to be more effective on larger instances. This demonstrates the advantage of GP-NSGA-III, especially on challenging problems. In addition, GP-NSGA-III appeared to be more effective at improving IGD than HV. This might be GP-NSGA-III seem to contribute most of the non-dominated solutions in the approximate Pareto-front.

4.5.3 Discussions

In this section, we further analyze the evolved dispatching rules and obtained distribution of the optimal solutions. We also show the average HV and IGD of the non-dominated solutions evolved by compared algorithms across multiple generations.

Computational Complexity of the Algorithm

The proposed algorithm (GP-NSGA-III) is hybridized, combines GP with the evaluation scheme of NSGA-III. So, the main computational complex-

Table 4.2: The mean and standard deviation over the HV values on the test instances of the compared algorithms.

ID	#J_#M	GP-NSGA-III	GP-NSGA-II	GP-SPEA2
1	15_15	**.0292(.0068)**	.0035(.0007)	.0170(.0028)
2	15_15	.0541(.0049)	**.3430(.0493)**	.0345(.0037)
3	15_15	**.0668(.0062)**	.0197(.0021)	.0179(.0017)
4	15_15	.0329(.0049)	**.1136(.0250)**	.0126 (.0012)
5	15_15	**.0344(.0034)**	.0122(.0012)	.0116(.0017)
6	20_15	.1027(.0087)	**.1796(.0094)**	.0410(.0032)
7	20_15	.0668(.0085)	**.2354(.0122)**	.0425(.0017)
8	20_15	.1193(.0118)	**.2506(.0138)**	.0457(.0029)
9	20_15	**.1287(.0109)**	.0593(.0035)	.0602(.0047)
10	20_15	**.1106(.0118)**	.0876(.0038)	.0533(.0032)
11	20_20	.0945(.0056)	**.2706(.0139)**	.0307(.0031)
12	20_20	.0873(.0064)	**.5793(.0209)**	.0361(.0021)
13	20_20	.0348(.0057)	.0332(.0015)	.0346(.0024)
14	20_20	**.0826(.0047)**	.0241(.0017)	.0330(.0025)
15	20_20	.0030(.0032)	**.0161(.0012)**	.0158(.0017)
16	30_15	**.2235(.0188)**	.1729(.0048)	.1263(.0062)
17	30_15	**.2092(.0132)**	.1514(.0046)	.1139(.0047)
18	30_15	**.1932(.0132)**	.1402(.0041)	.1194(.0057)
19	30_15	**.1976(.0161)**	.1525(.0057)	.1058(.0040)
20	30_15	**.1863(.0157)**	.1533(.0044)	.1163(.0046)
21	30_20	**.1863(.0184)**	.1188(.0038)	.0839(.0047)
22	30_20	**.0777(.0067)**	.0230(.0019)	.0249(.0031)
23	30_20	**.1739(.0127)**	.1119(.0034)	.0880(.0058)
24	30_20	**.1687(.0147)**	.1401(.0028)	.0900(.0034)
25	30_20	**.1692(.0127)**	.1252(.0035)	.0920(.0044)
26	50_15	**.2609(.0139)**	.1999(.0062)	.1713(.0070)
27	50_15	**.3092(.0229)**	.2035(.0047)	.1750(.0061)
28	50_15	**.2453(.0159)**	.1820(.0070)	.1577(.0058)
29	50_15	**.2811(.0216)**	.1789(.0062)	.1578(.0067)
30	50_15	**.2708(.0177)**	.1845(.0053)	.1599(.0098)
31	50_20	**.2143(.0135)**	.1738(.0062)	.1366(.0054)
32	50_20	**.3182(.0193)**	.2001(.0042)	.1582(.0045)
33	50_20	**.2568(.0162)**	.1829(.0066)	.1383(.0051)
34	50_20	**.2955(.0189)**	.1876(.0046)	.1522(.0055)
35	50_20	**.2737(.0130)**	.1851(.0043)	.1500(.0064)
36	100_20	**.3333(.0163)**	.2169(.0062)	.2131(.0081)
37	100_20	**.2687(.0164)**	.0896(.0037)	.0796(.0048)
38	100_20	**.2659(.0145)**	.2141(.0057)	.2132(.0061)
39	100_20	**.2345(.0134)**	.1141(.0668)	.1111(.0055)
40	100_20	**.3564(.0190)**	.2217(.0046)	.2211(.0059)

Table 4.3: The mean and standard deviation over the IGD values on the test instances of the compared algorithms.

ID	#J_#M	GP-NSGA-III	GP-NSGA-II	GP-SPEA2
1	15_15	**.0071(4.0E-05)**	.0077(4.9E-05)	.0087(1.4E-05)
2	15_20	.0071(6.9E-05)	**.0037(2.1E-05)**	.0105(9.3E-05)
3	15_15	**.0009(1.9E-05)**	.0018(6.1E-06)	.0017(1.8E-05)
4	15_15	.0057(3.9E-05)	**.0036(1.2E-05)**	.0119(4.7E-05)
5	15_15	**.0013(1.6E-05)**	.0023(1.4E-05)	.0037(4.9E-05)
6	20_15	**.0024(2.2E-05)**	.0028(3.3E-05)	.0051(6.2E-05)
7	20_15	.0110(7.5E-05)	**.0088(5.9E-05)**	.0167(1.4E-05)
8	20_15	**.0027(4.0E-05)**	.0038(3.3E-05)	.0060(9.2E-05)
9	20_15	**.0023(2.3E-05)**	.0033(3.9E-05)	.0054(7.5E-05)
10	20_15	**.0029(3.0E-05)**	.0044(5.6E-05)	.0072(9.0E-05)
11	20_20	**.0017(1.4E-05)**	.0025(1.5E-05)	.0046(6.3E-05)
12	20_20	**.0023(1.8E-05)**	.0031(2.4E-05)	.0048(5.9E-05)
13	20_20	**.0016(9.7E-06)**	.0027(1.0E-05)	.0042(8.1E-05)
14	20_20	**.0021(2.3E-05)**	.0027(2.2E-05)	.0045(5.7E-05)
15	20_20	**.0032(3.6E-05)**	.0050(4.0E-05)	.0083(9.3E-05)
16	30_15	**.0009(1.7E-05)**	.0013(1.9E-05)	.0020(2.5E-05)
17	30_15	**.0024(3.3E-05)**	.0032(4.9E-05)	.0054(5.5E-05)
18	30_15	**.0017(2.0E-05)**	.0023(3.0E-05)	.0036(3.4E-05)
19	30_15	**.0025(3.4E-05)**	.0033(3.8E-05)	.0052(6.3E-05)
20	30_15	**.0015(2.3E-05)**	.0018(3.1E-05)	.0030(2.8E-05)
21	30_20	**.0026(4.1E-05)**	.0033(4.7E-05)	.0050(1.0E-05)
22	30_20	**.0040(2.5E-05)**	.0048(2.7E-05)	.0080(7.5E-05)
23	30_20	**.0027(2.0E-05)**	.0031(3.1E-05)	.0048(4.4E-05)
24	30_20	**.0033(5.1E-05)**	.0037(5.0E-05)	.0059(6.3E-05)
25	30_20	**.0030(4.0E-05)**	.0037(3.6E-05)	.0059(6.1E-05)
26	50_15	**.0013(1.3E-05)**	.0018(2.9E-05)	.0031(3.5E-05)
27	50_15	**.0007(1.2E-05)**	.0013(2.2E-05)	.0022(2.7E-05)
28	50_15	**.0041(4.2E-05)**	.0047(8.3E-05)	.0081(9.8E-05)
29	50_15	**.0017(3.0E-05)**	.0023(4.2E-05)	.0041(4.9E-05)
30	50_15	**.0020(3.6E-05)**	.0024(4.0E-05)	.0041(4.6E-05)
31	50_20	**.0010(1.0E-05)**	.0015(4.2E-05)	.0024(4.3E-05)
32	50_20	**.0008(9.6E-06)**	.0013(2.1E-05)	.0021(3.2E-05)
33	50_20	**.0033(5.0E-05)**	.0035(4.5E-05)	.0058(6.6E-05)
34	50_20	**.0012(2.4E-05)**	.0016(2.8E-05)	.0026(2.7E-05)
35	50_20	**.0013(2.2E-05)**	.0020(3.3E-05)	.0034(3.4E-05)
36	100_20	**.0011(1.6E-05)**	.0016(1.9E-05)	.0030(3.8E-05)
37	100_20	**.0005(8.4E-06)**	.0012(1.9E-05)	.0010(1.4E-05)
38	100_20	**.0015(1.7E-05)**	.0018(3.3E-05)	.0035(8.8E-05)
39	100_20	**.0012(1.3E-05)**	.0027(3.7E-05)	.0047(5.8E-05)
40	100_20	**.0009(1.4E-05)**	.0015(2.3E-05)	.0028(4.6E-05)

ity resulted from an evaluation scheme of NSGA-III and computational time for evolving dispatching rules. The evaluation scheme of NSGA-III holds a computational complexity of $\mathcal{O}(MN^2)$ [43], where M is the objective number and N is the population size. In addition, the computational complexity of two compared algorithms (NSGA-II and SPEA2) are also having $\mathcal{O}(MN^2)$ [44, 211].

To know the computational cost of each algorithm to evolve rules, we performed several experiments. Figure 4.5 shows each algorithm's computational cost (GP-NSGA-III, GP-NSGA-II, GP-SPEA2) to evolve rules. Figure 4.5 reveals that the computational cost of GP-NSGA-III is lower than GP-NSGA-II and GP-SPEA2. In addition, the computational time of GP-NSGA-III is quite consistent in all generations as compared to GP-NSGA-II and GP-SPEA2. One possible explanation is that GP-NSGA-II and GP-SPEA2 have trouble finding effective compact rules in most generations and good performances can only be obtained by large and sophisticated rules in the GP-NSGA-II and GP-SPEA2 populations. As these rules are evolved, they will take GP-NSGA-II and GP-SPEA2 more time for evaluations.

The lengths of rules obtained by GP-NSGA-III, GP-NSGA-II, and GP-SPEA2 are shown in Figure 4.6. The length of a rule is the total number of nodes, including terminal and function nodes. In most generations, GP-NSGA-III produces shorter rules as compared to GP-NSGA-II, and GP-SPEA2. This is also consistent with our results that the computational time of GP-NSGA-III is lower than GP-NSGA-II and GP-SPEA2. This is because GP-NSGA-III can find more compact rules which require less time for evaluations. Further, the rules generated by GP-SPEA2 is relatively short as compared to GP-NSGA-II. Given the poor performance of the rules generated by GP-SPEA2 in terms of HV and IGD (see Section 4.5.2), the results here show that GP-SPEA2 has failed to develop effective rules.

Figure 4.7 shows the length of the best rules of each run. Overall the

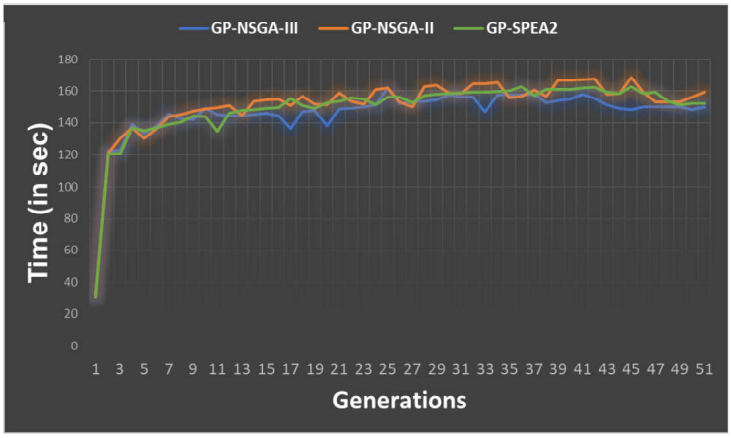


Figure 4.5: Computational time to evolve dispatching rules

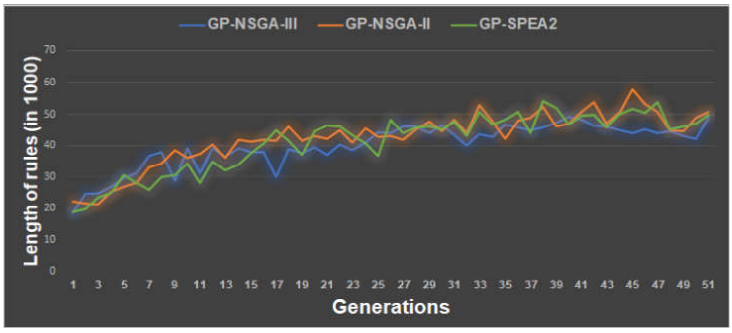


Figure 4.6: Length of rules of each generation in GP-NSGA-II, GP-SPEA2, and GP-NSGA-III.

the length of the best rules of GP-NSGA-II and GP-SPEA2 are quite long as compared to GP-NSGA-III which may directly affect the computational time of the algorithm.

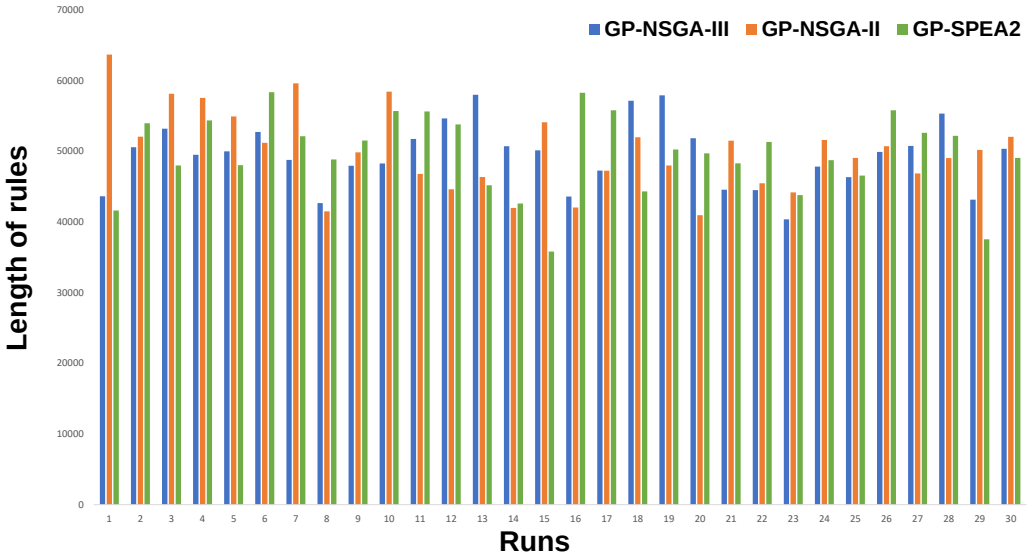


Figure 4.7: Length of the best rules of each run in GP-NSGA-II, GP-SPEA2, and GP-NSGA-III.

Analysis of dispatching rules

This section analyzes each terminal's frequency that was counted in the set of 30 independent runs of evolved rules from GP-NSGA-II, GP-SPEA2, and GP-NSGA-III. From the frequency of terminals, we can further analyze the relevant and irrelevant terminals. The bar chart in Figure 4.8 shows the frequency of terminals in each algorithm. We can be seen from Figure 4.8 that W and PT are the most frequently chosen terminals in all algorithms. However, we need to investigate the relevance of terminals to optimize tardiness and flowtime objectives.

According to the existing studies in the literature [70, 114], MRT, PT, WKR, WINQ, NOINQ, FDD, and NOPT are useful terminals (relevant) for optimizing flowtime objectives. Specifically, PT, WINQ, and WKR are the most important three terminals for optimizing the flowtime objective. On the other hand, WINQ, NOINQ, NOPT, W, PT, MRT, and DD are the

useful terminals for optimizing tardiness objectives. PT, DD, and W are the most useful terminals for optimizing tardiness objectives.

It can be seen from Figure 4.8 that W, PT are the most frequently chosen terminals in all algorithms. The frequency of occurrences of PT and W terminals is higher in the GP-NSGA-III than GP-NSGA-II. It can also be seen that WINQ, MRT, NOPT, and DD are also more frequent terminals in GP-NSGA-III. NOINQ and WKR are more frequent terminals in GP-NSGA-II. GP-NSGA-III and GP-NSGA-II have an identical frequency of occurrences of FDD. It is not very clear whether the remaining terminals (WIQ, NMRT) are useful or not.

It can be seen from Figure 4.8 that GP-NSGA-III has a greater frequency of useful terminals which positively affects the performance of the GP-NSGA-III. As mentioned in the last section 4.5.2, GP-NSGA-III evolved significantly better rules as compared to the other algorithms in terms of HV and IGD. Therefore, GP-NSGA-III selects well-converged and well-diversified rules. In this sense, these selected rules are well-optimized and have more occurrences of these useful terminals.

Feature relevance

The contribution of a feature (terminal) to a priority function on a set of JSS instances show the relevancy of each terminal. In other words, the terminal (t_i) contribution to the priority function is the difference between the fitness values before and after removing t_i from the priority function. A positive value of contribution indicates that after removing t_i, the new dispatching rule will generate worse schedules for the tested JSS instances. On the other hand, a negative value implies that t_i makes a negative contribution to a priority function, and removing t_i can lead to an improvement in the performance. If the value is zero, t_i makes no contribution. Intuitively, if t_i is a more relevant or useful terminal, it contributes to the rules that perform well in tested JSS instances.

To perform this experiment, we are using an offline terminal-selection

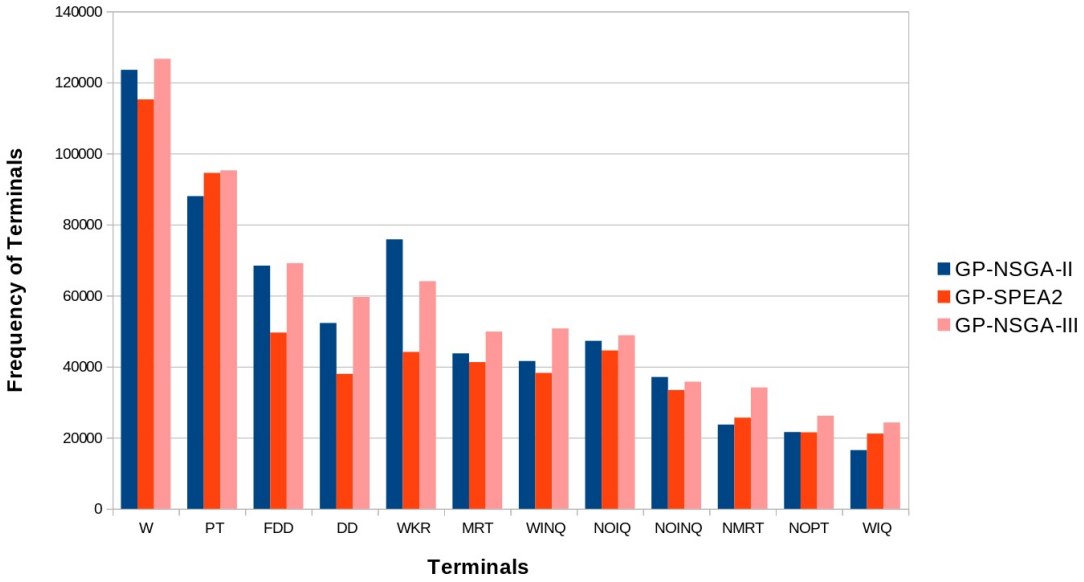


Figure 4.8: Frequency of terminals in GP-NSGA-II, GP-SPEA2, and GP-NSGA-III.

strategy. We used the following steps for the selection of terminals.

1. simplify the obtained rules(see Section 4.5.2) according to the principles which were proposed in [131],

2. calculate the relevance of each terminal which is based on the 30 best rules.

The bar chart in Figure 4.9 shows the frequency of each terminal (after simplification of rules) that was counted in the set of 30 independent runs of evolved rules from GP-NSGA-III, GP-NSGA-II, GP-SPEA2. Figure 4.8 reveals that W and PT are the most frequently chosen terminals in all algorithms which is consistent with Figure 4.8 where the W and PT are also the most frequent terminals. Figure 4.9 also shows the PT and W terminals' frequencies are higher in the GP-NSGA-III than GP-NSGA-II and GP-SPEA2. It can be seen from Figure 4.8 that the WKR is a more frequent terminal in GP-NSGA-II but Figure 4.9 shows after simplification

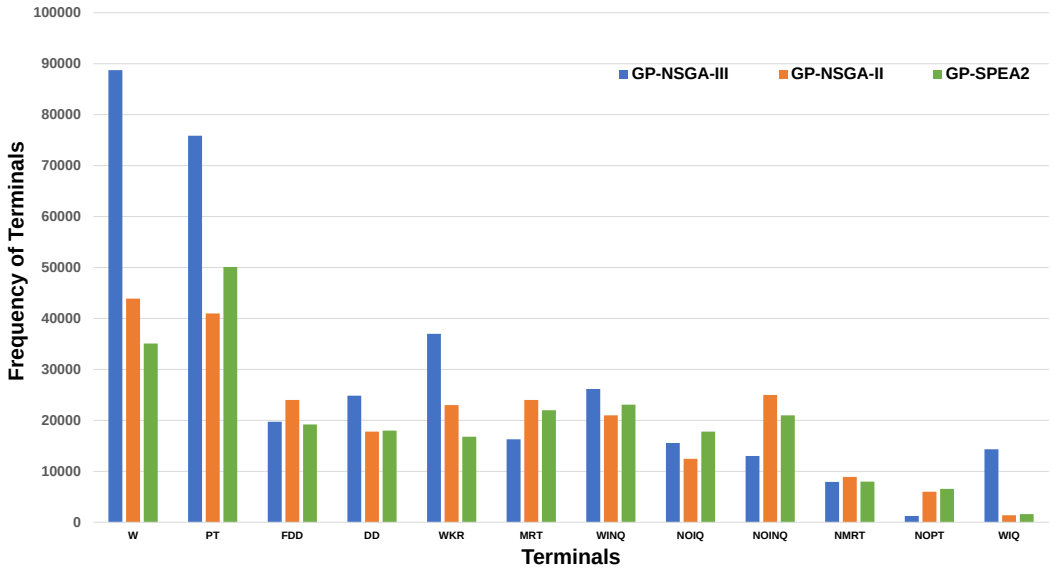


Figure 4.9: Frequency of terminals (after simplification) in GP-NSGA-III, GP-NSGA-II, GP-SPEA2.

of rules, WKR is the most frequent terminals in GP-NSGA-III. Moreover, GP-NSGA-III and GP-NSGA-II have an identical frequency of occurrences of FDD in Figure 4.8 but after simplification of rules, FDD is higher in the GP-NSGA-II.

Figure 4.10 shows the average relevance of terminals from all three algorithms to optimize tardiness and flowtime objectives. Figure 4.10 reveals that PT has the largest relevance (nearly 0.48), followed by PT, which is about 0.4. There are 4 features whose relevance is almost close to zero (FDD, NOIQ, NMRT, and WIQ), indicating that they made no contribution to the best rule most of the time. Hence W, PT, DD, WKR, MRT, WINQ, NOINQ, and NOPT are selected as relevant terminals to tardiness and flowtime objectives. Most of these terminals are higher in the GP-NSGA-III than GP-NSGA-II and GP-SPEA2 which can positively affect the performance of the GP-NSGA-III.

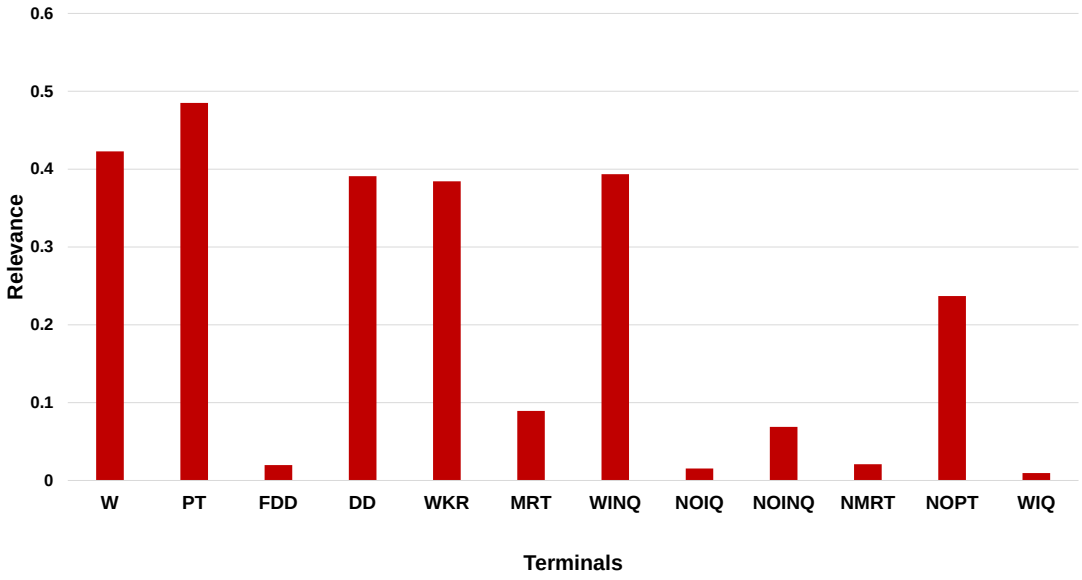


Figure 4.10: The average relevance of each terminal over the 30 independent runs of GP-NSGA-III, GP-NSGA-II and GP-SPEA2 on the training set.

Further analysis

To further investigate how the reference point scheme affects the GP search process, the average HV and average IGD of the non-dominated solutions obtained on training instances are plotted for each generation during the 30 independent runs of the three compared algorithms, as given in Figures 4.11 and 4.12.

Figures 4.11 and 4.12 show that GP-NSGA-III obtained better convergence curves in terms of both HV and IGD as compared to the GP-NSGA-II and GP-SPEA2. The convergence curve shows that in terms of HV, GP-NSGA-II performs better in the early generations but then GP-NSGA-III has better HV value than the other two compared algorithms. GP-SPEA2 obtained the worst performance in both HV and IGD, which is consistent with the results reported in Table 4.1.

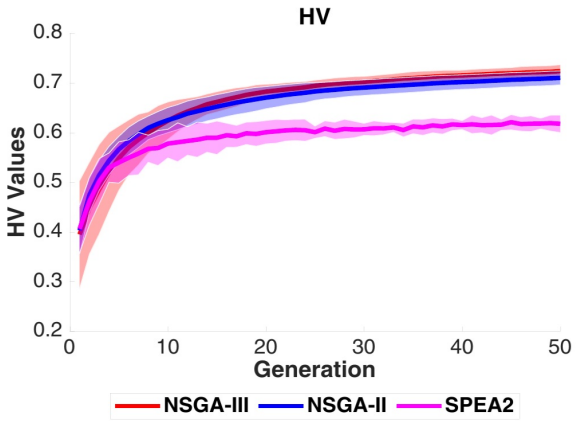


Figure 4.11: The average HV value of the non-dominated solutions on the training set during the 30 independent GP runs.

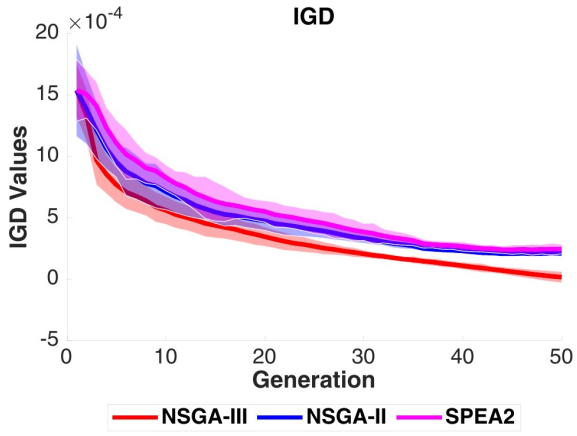
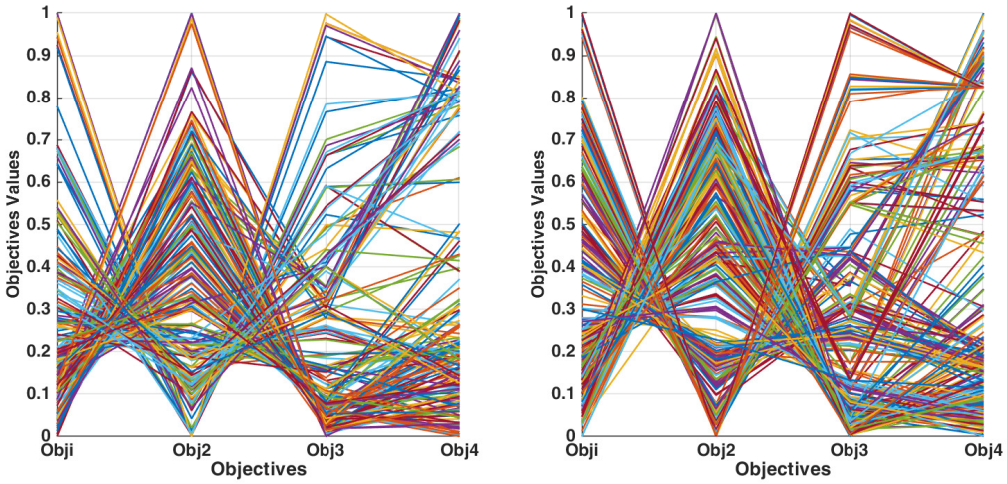


Figure 4.12: The average IGD value of the non-dominated solutions on the training set during the 30 independent GP runs.

Figures 4.13 (a) and 4.13 (b) depict the populations' distribution of the fitness values of GP-NSGA-II and GP-NSGA-III in generations 50 with 4 objectives. It can be seen that GP-NSGA-III managed to achieve better values in objectives 3 and 4 and covered a wider range of trade-offs corresponding to objective 1. This also shows that GP-NSGA-III can evolve



(a) GP-NSGA-II (b) GP-NSGA-III.

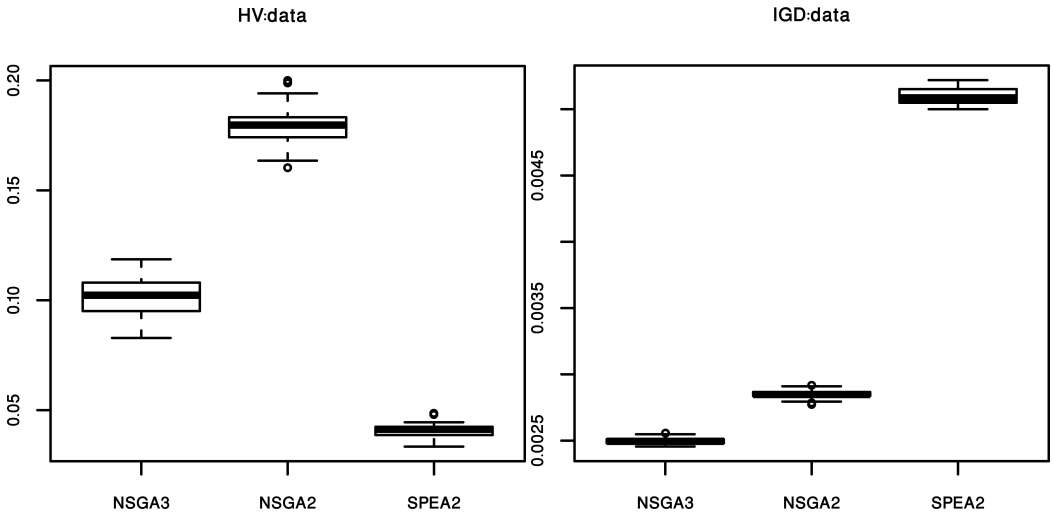
Figure 4.13: Parallel coordinate plot for the distribution of the reference points and the fitness values of the population at generations 50 .

more diversified rules than GP-NSGA-II.

Figures 4.14 (a) to 4.16 (b) represents the box plots of the HV and IGD values on different test instances. For the HV metric, GP-NSGA-III achieved significantly better results on complex instances (ID-21 and ID-32). On the other hand, GP-NSGA-II performs significantly better on the smallest instance (ID-6). As for IGD, GP-NSGA-III has again proven to be significantly better than the other two algorithms on both small and complex instances.

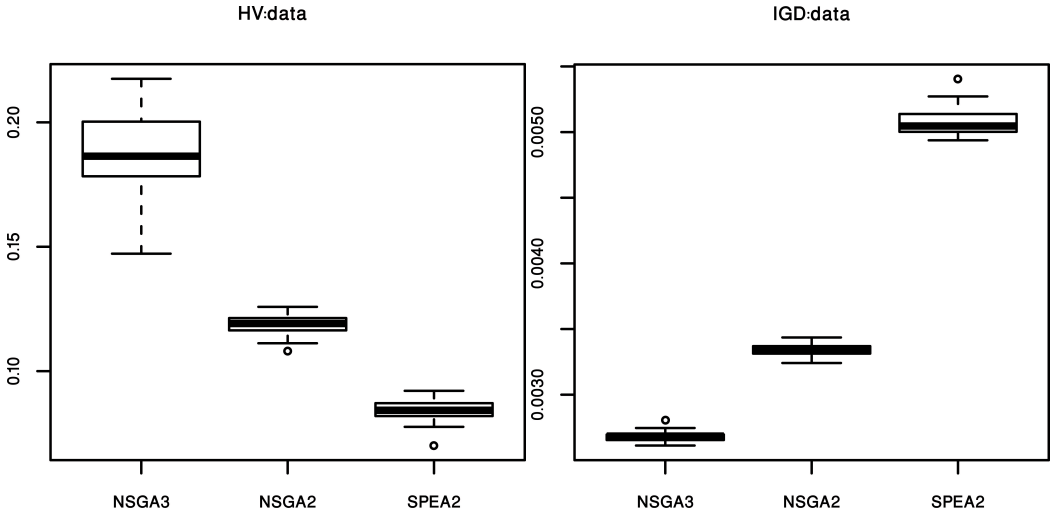
4.6 Chapter summary

Dispatching rules for JSS problems are mainly designed in the literature for single-objective and multi-objective optimization problems. However, several industries (e.g., manufacturing and cloud) require suitable dispatching rules that can balance several objectives instead of simply focus-



(a) HV (b) IGD.

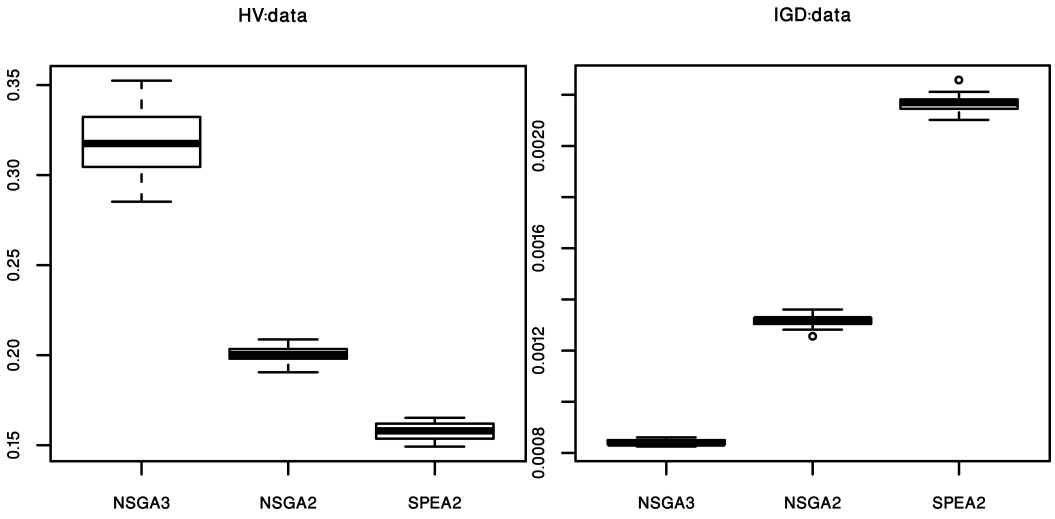
Figure 4.14: Box plots on instance 6 of the compared algorithms.



(a) HV (b) IGD.

Figure 4.15: Box plots on instance 21 of the compared algorithms.

ing only on one objective. Thus, this chapter studies the importance of considering many (four) potentially conflicting objectives for effective JSS. Aiming at solving JSS problems with many objectives, the new algorithm



(a) HV (b) IGD.

Figure 4.16: Box plots on instance 32 of the compared algorithms.

was developed that seamlessly combines GP-HH for evolving dispatching rules with the selection technique introduced in NSGA-III.

Extensive experiments have been performed to investigate the proposed algorithm's effectiveness compared with the other two popular multi-objective algorithms, i.e., NSGA-II and SPEA2. The experimental studies utilized the Taillard static job-shop benchmark set. Our experiment results gave evidence that the proposed algorithm performed consistently better than GP-NSGA-II and GP-SPEA2 on the majority of the problem instances.

Several interesting conclusions were also drawn. The results have shown on both training and test instances, GP-NSGA-III can evolve more effective rules than GP-NSGA-II and GP-SPEA2 in terms of HV and IGD. Further, GP-NSGA-III has well-optimized rules; therefore, it has more effective terminals than GP-NSGA-II and GP-SPEA2. Further, the results show that the evolved rules enjoy high generalization capability and can effectively tackle both training instances and unseen problem instances.

In the next chapter, we will focus on addressing the second issue: di-

versity maintenance [199]. In the literature, diversity maintenance is often controlled by providing multiple predefined reference points. These points are widely distributed. However, uniformly distributed reference points are inappropriate when the true Pareto-front is non-uniform and irregular [85, 86], such as JSS problems. These problems may have some reference points that are never associated with any of the Pareto-optimal solutions and will become useless reference points during evaluation. The existence of these useless reference points in algorithms significantly affects its performance. To address this issue, we have proposed a new reference point adaptation mechanisms in our next chapter.

Chapter 5

Reference Points Adaptation for Many-Objective JSS

5.1 Introduction

This chapter addresses the second issue of many-objective optimization problems (MaOPs), which is diversity maintenance [199]. In the literature, diversity maintenance is often controlled by providing multiple predefined reference points such as NSGA-III [43] and SPEAR [92]. These points are widely distributed on a normalized hyperplane (simplex) [31, 43] and they are expected to guide the population to be distributed evenly over the true Pareto-front [43].

Reference points based algorithms such as NSGA-III [43], RVEA [31], and SPEAR [92] use a predefined set of uniformly distributed reference points and they have successfully solved various practical many-objective optimization problems [43]. This set of widely distributed reference points promotes the diversity of evolved solutions in the Pareto-front with three to fifteen objectives.

Even though these algorithms have successfully solved various practical MaOPs, they still have challenges when applying to real-world problems such as the car cab design problem and the design of vehicle prob-

lems. These real-world problems usually have non-uniform and irregular Pareto-front. Therefore, the adoption of uniformly distributed reference points affects algorithms' performance adversely [85, 86]. This is because many of these reference points are never associated with any of the optimal solutions and become useless. Also, useless reference points will notably affect the performance of reference points based algorithms which is hard to achieve whenever the true Pareto-front is irregular (e.g., non-convex or non-uniform). This issue has been highlighted in the literature [85, 86].

Particularly, in problems with an irregular, non-uniform and disconnected Pareto-front, a few reference points are associated with more than one optimal solution in their closest proximity. If the reference point is associated with many solutions, they may not be selected during evolution. Therefore, candidate solutions that should enjoy higher selection opportunities may not be selected because they are associated with a popular reference point [86]. Figure 5.1 shows that reference point F is associated with many solutions than the other useful points (A, C, and D). Consider the population size (N=10) in this example. Therefore, reference point (A or C or D) has the minimum associated solutions selected first and members associated with these points become a part of the next generation. The reference point F has a lesser selection opportunity than the other reference points. This is because a number of solutions associated with A, C, and D are equal to the population size. Thus, solutions associated with F may not be part of the next generation and highly affect the solution diversity.

In many combinatorial optimization problems, such as job shop scheduling (JSS) problems, the true Pareto-front is usually irregular and discontinuous [142]. JSS problems have also inherited the issue of useless reference points, i.e., the solution diversity's demotion. Despite the promising results of GP-NSGA-III (the detailed information can be seen in Subsection 4.3.2 of Chapter 4), we have found (shown in Figure 5.2) that

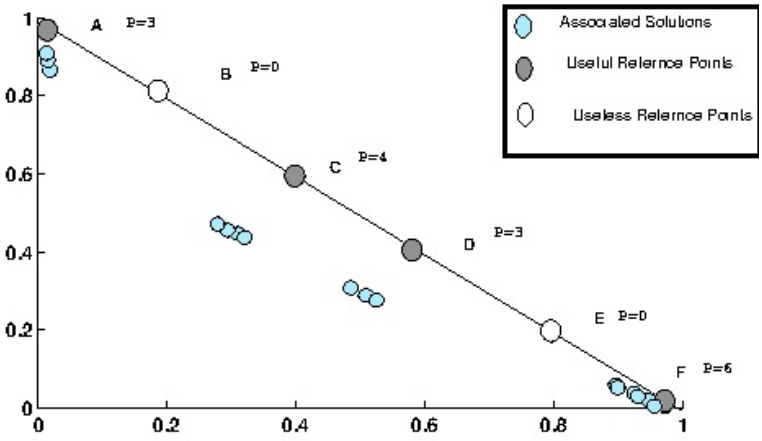


Figure 5.1: Associated and contributing solutions on a convex curvature of Pareto-front

many reference points are useless. They are never associated with any dispatching rules on the evolved Pareto-front in JSS. It can also be seen in Figure 5.2 that the number of useless references points in GP-NSGA-III kept increasing from *910* to about *980*. Suppose only a few reference points are truly associated with the Pareto-optimal dispatching rules at the current generation. In that case, it is not easy to distinguish and select these rules to improve diversity for future generations. Due to the above reason, we should find how to get better matches between reference points and the evolved Pareto-front that can help to enhance solution diversity and, therefore, the performance of GP-NSGA-III.

To enhance the solutions' diversity and reduce the useless points, the distribution of reference points must have a better match with the candidate solutions' distribution. For this purpose, A-NSGA-III [85], RVEA* [31] proposed reference points adaptation strategies for generating reference points according to the approximated Pareto-front which can be seen in Section 2.1.6 of Chapter 2.

RVEA* and A-NSGA-III mechanisms require the number of refer-

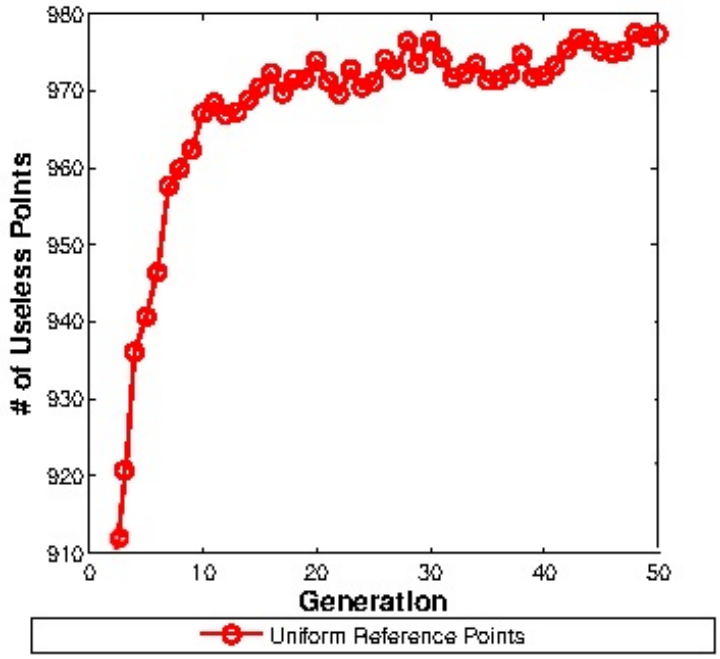


Figure 5.2: The curve of average number of useless reference points in GP-NSGA-III on the training instances during the 30 independent GP runs

ence points to be changed dynamically. In high-dimensional objective space, when the number of reference points becomes large, adding or replacing reference points may affect algorithms' efficiency. Moreover, both RVEA* and A-NSGA-III use some heuristics for generating reference points. These heuristics do not have any concrete knowledge of the objective space and generated the reference points based on the last generation solutions.

This chapter aims to develop *two new effective mechanisms (model-free and model-driven) for reference point generation* and improve the association between reference points and the Pareto-front during evaluation to address the issues mentioned above. Furthermore, better matches between reference points and the evolved Pareto-front can help to enhance the solution diversity and quality of dispatching rules. The objective of this chapter is further broken down into two major sub-objectives:

1. To develop a new adaptive reference point strategy using the model-free approach (heuristics approach). In the model-free approach, we proposed a new adaptation mechanism inspired by particle swarm optimization (PSO). This reference point adaptation mechanism is expected to significantly improve our proposed algorithm's performance by reducing useless reference points. The proposed approach will be evaluated on the Taillard benchmark JSS problem and is compared with the baseline algorithm (GP-NSGA-III).

2. To develop an adaptation mechanism using a model-based technique that estimates the density of solutions from each defined sub-location in a whole objective space. Subsequently, it applies a Gaussian Process on the model, which gives smoothness to the model. The performance of the proposed adaptive model-based technique is also verified on JSS instances and compared with GP-A-NSGA-III [85] and GP-NSGA-III. In this section, we will further show the general applicability and usefulness of the model-based reference point adaptation technique by applying them to the benchmarks problems.

This chapter is divided into two major parts. First, we provide the model-free reference points generation algorithm, followed by the experimental design and the results compared with GP-NSGA-III approaches. Afterward, the model-based algorithms in the context of adaptive reference point generation are discussed. Next, we present the experimental design and the results. Finally, a chapter summary is provided.

5.2 General framework for adaptive reference points generation

The proposed algorithms for generating adaptive reference points can be considered as a major enhancement of GP-NSGA-III (GP-NSGA-III is de-

veloped in Section 4.3.2 of Chapter 4). Both model-based and model-free mechanisms utilize the adaptive reference point's general framework, which is similar to that of the GP-NSGA-III. In GP-NSGA-III, the population is created after the niching operation P_{t+1}. Further, the niche counts ρ_i (the number of population members associated with ith reference point, $z_i \in \boldsymbol{Z}$) for each reference point is updated then reference points are generated adaptively. More details of adaptive reference point mechanisms will be discussed in the next two sections.

5.3 Model-free adaptive reference points generation

In GP-NSGA-III, the size of reference points is almost equal to the population's size (N). Additionally, in an ideal condition, every reference point is expected to be associated with one population member. Thus, if $\rho \geq 2$, some other reference point has ρ (niche count) equal to zero and is considered to be a useless point.

In this model-free approach, each reference point has its fitness value which is defined as the number of individuals in the population associated with reference points. In other words, the fitness value of the useless reference point has a niche count (ρ) which is equal to zero. These useless reference points are relocated to those reference points which have maximum fitness value. This relocation strategy provides a better match between reference points and the Pareto-front evolved by GP-HH. This is expected to help to enhance the solution diversity and, therefore, the performance of GP-NSGA-III.

PSO has been proven in the literature to be highly effective for approximating arbitrary distributions such as the fitness landscape [153, 204]. Our model-free approach GP-Adaptive-NSGA-III(PSO). (GP-A-NSGA-III(PSO)) is inspired by the principle of PSO, which has a swarm of parti-

cles (reference points). The particle's size is similar to the size of the reference point (nref). Particularly, each reference point is seen as a particle in the swarm and each reference point consists of its own velocity vector $\boldsymbol{V}$ along with the position vector $\boldsymbol{X}$.

Like the PSO, GP-A-NSGA-III(PSO) also has the concept of the global best particle. The global best particle is the one with the most individuals associated with it. The swarm consists of only one single best position in the PSO, but our proposed algorithm can have more than one *global best* with no elitism strategy being adopted for the global best. This can be seen in Algorithm 5 (line 4). This way, the reference points can have sufficient diversity. Every useless reference point can be optionally attracted to its closest global best location. This GP-A-NSGA-III(PSO) reference point update scheme (update($\boldsymbol{Z}$)) is described in Algorithm 5 where $\boldsymbol{Z}$ is a set of reference points.

Each useless particle updates its position according to the new velocity shows in line 11 in Algorithm 5. Note that when updating the velocity (line 9) in Algorithm 5, the term for the local best position (PSO uses of memory of each particle's best location) is ignored. It can be seen that the fitness of particles (reference points) depends on the distribution of the whole swarm and the positions of other particles. Thus, it may not be meaningful to move towards the local best, which can become worse upon particles' movements prevent the majority of reference points from converging to small areas in the objective space. The new velocity is defined as

$$\boldsymbol{V}_{\boldsymbol{i,j}}^{\boldsymbol{g}} = w * \boldsymbol{V}_{i,j}^{g-1} + c_2 * rand() * (\boldsymbol{Z}^* - \boldsymbol{X}_{i,j}^{g-1}). \tag{5.1}$$

where Z^* denotes the global best position and the parameters w, c_2 are inertia weight and positive constants respectively. In the baseline, PSO algorithm w is chosen as [0.5 0.9]. Also, $\boldsymbol{V}_{i,j}^{g-1}$ is a velocity vector of i-th element of the preceding generation, which includes the M objective values.

Algorithm 5: Update of the reference points.

Input : Reference points $\boldsymbol{Z} = (\boldsymbol{X}, \boldsymbol{V})$, ρ
Output: Updated reference points $\boldsymbol{Z}^g$

Calculate fitness $fit(\boldsymbol{Z}_i)$= $\boldsymbol{\rho_i}$ (number of individuals associated for each $\boldsymbol{Z}_i \in \boldsymbol{Z}$);
Calculate $w = w_{max} - g \cdot (w_{max} - w_{min})/g_{max}$;
; // inertia weight of generation(g), g_{max} is the maximum generation.
for $i = 1 \rightarrow nref$ **do**
    Calculate the global best $\boldsymbol{Z_k}^* = \arg\max\{fit(\boldsymbol{Z_i})\}$; // global best
end
for $i = 1 \rightarrow nref$ **do**
    if $\boldsymbol{\rho}_i \neq 0$ **then**
        for $j = 1 \rightarrow M$ **do**
            $\boldsymbol{V_{i,j}^g} = w * \boldsymbol{V}_{i,j}^{g-1} + c_2 * rand() * (\boldsymbol{Z_k}^* - \boldsymbol{X}_{i,j}^{g-1})$; // velocity vector of generation g.
            ;
            $\boldsymbol{X}_{i,j}^g = \boldsymbol{X}_{i,j}^{g-1} + \boldsymbol{V}_{i,j}^g$; // position vector of generation g.
            ;
        end
    end
end
return $\boldsymbol{Z}^g = (\boldsymbol{X}^g, \boldsymbol{V}^g)$;

5.3.1 Design of experiment

The experimental design is carried out to evaluate the GP-A-NSGA-III (PSO) and GP-NSGA-III approaches. In the experimental studies, we compared the performance of GP-A-NSGA-III(PSO) with the baseline GP-NSGA-III. The JSS benchmark, the Taillard (TA) static JSS benchmark instances (see in Section 3.3 of Chapter 3) is selected as a testbed. In the experiments, we considered four potentially conflicting objectives:the mean flowtime (mF) (see equation (2.1) of Chapter 2) , maximal flowtime (maxF) (see equation (2.2) of Chapter 2), mean weighted tardiness (mWT) (see

equation (2.6) of Chapter 2) and maximal weighted tardiness (maxWT) (see equation (2.7) of Chapter 2).

The terminal set and function set are described in Table 3.2 of Chapter 3. Both GP-A-NSGA-III(PSO) and GP-NSGA-III adopt the GP representation (tree-based) and evolutionary operators (e.g., initialization, crossover, and mutation). For both compared algorithms, the population size is set to 1025. The crossover, mutation and reproduction rates are set similar to Subsection 4.4.1 of Chapter 4. The maximal number of generations (g_{max}) is set to 51. For the PSO parameters, we set $c_2 = 2$, $w_{min} = 0.4$ and $w_{max} = 0.9$, which are standard settings used in many existing studies [100]. In the experiments, we again use IGD [206] and HV [212] to compare the algorithms (The detailed information can be seen in Section 3.3 Chapter 3).

5.3.2 Results and discussions

During the GP search process, a rule is evaluated on the 40 training instances. The fitness function for each objective is defined as the average normalized objective value of the schedule obtained by applying that rule to each of the 40 training instances. For each algorithm, 30 GP runs obtained 40 final dispatching rules. Then, the rules were tested on the 40 test instances.

Overall results

Table 5.1 shows the mean and standard deviations of the test performance (HV and IGD) of the rules obtained by GP-NSGA-III and GP-A-NSGA-III(PSO). In addition, for each test instance, the *Wilcoxon rank-sum* test with the significance level of 0.05 was conducted separately on both the HV and IGD of the rules obtained by the two compared algorithms. That is, if the p-value is smaller than 0.05, then the best algorithm is considered significantly better than the other algorithm. The significantly better results are marked in bold.

Table 5.1: The mean and standard deviation over the HV and IGD values on the test instances of the compared algorithms in the 4-obj experiment.

Problem Instances		HV		IGD	
ID	#J_#M	GP-NSGA-III	GP-A-NSGA-III(PSO)	GP-NSGA-III	GP-A-NSGA-III(PSO)
1	15_15	.0096(.0145)	**.2414(.0368)**	.0265(.0008)	**.0088(.0113)**
2	15_15	.1600(.0182)	**.1666(.0110)**	.0254(.0005)	**.0218(.0004)**
3	15_15	.0806(.0125)	**.0906(.0127)**	**.0205(.0008)**	.0292(.0006)
4	15_15	**.1263(.0183)**	.0692(.0091)	**.01765(.0007)**	.0218(.0006)
5	15_15	.1661(.0202)	.1752(.0192)	**.0190(.0005)**	.0237(.0007)
6	20_15	.1271(.0176)	**.1421(.0139)**	**.0154(.0004)**	.0211(.0004)
7	20_15	.2488(.0683)	**.2538(.0695**)	.0079(.0013)	.0077(.0014)
8	20_15	.1115(.0201)	**.2015(.0253)**	.0186(.0006)	.01819(.0026)
9	20_15	.1700(.0148)	**.1839(.0219)**	.0170(.0005)	**.0149(.0005)**
10	20_15	.1086(.0151)	**.3580(.0366)**	**.0156(.0002)**	.0160(.0003)
11	20_20	.0114(.0038)	**.1087(.0118)**	.0299(.0006)	**.0240(.0008)**
12	20_20	.0852(.0133)	**.1206(.0134)**	**.0173(.0004)**	.0222(.0006)
13	20_20	.1540(.0151)	.1569(.0326)	.0188(.0003)	**.0137(.0002)**
14	20_20	.0504(.01103)	**.0704(.0118)**	.0328(.0008)	**.0272(.0007)**
15	20_20	**.3985(.0225)**	.2454(.0199)	**.0109(.0004)**	.0150(.0002)
16	30_15	.2465(.030)	.2375(.0234)	**.0144(.0006)**	.0140(.0002)
17	30_15	.1813(.0096)	**.2278(.0229)**	.0138(.0003)	**.0097(.0004)**
18	30_15	**.3198(.0233)**	.1957(.0132)	**.0131(.0004)**	.0152(.0003)
19	30_15	.2789(.0126)	**.3004(.0197)**	.0150(.0004)	**.0114(.0004)**
20	30_15	**.2575(.0312)**	.2124(.0312)	**.0144(.0005)**	.0195(.0003)
21	30_20	.1347(.0657)	**.2325(.0792)**	.0135(.0022)	**.0098(.0014)**
22	30_20	.2365(.0472)	**.3027(.0470)**	.0065(.0010)	**.0052(.0006)**
23	30_20	.2944(.0398)	.2984(.0410)	.0046(.00004)	.0045(.0005)
24	30_20	.3812(.0503)	**.6161(.0174)**	.0070(.0009)	**.0018(.0004)**
25	30_20	.5199(.0477)	.5290(.0396)	.0059(.0012)	**.0046(.0005)**
26	50_15	.4563(.0417)	**.4872(.0270)**	.0051(.0009)	**.0039(.0004)**
27	50_15	.5710(.0361)	.5685(.0304)	.0040(.0009)	**.0032(.0003)**
28	50_15	.4598(.0398)	**.4966(.0250)**	.0049(.0010)	**.0036(.0003)**
29	50_15	.4862(.0372)	**.5125(.0251)**	.0045(.0007)	**.0037(.0003)**
30	50_15	.4510(.0406)	**.4732(.0240)**	.0033(.0005)	**.0026(.0002)**
31	50_20	.5085(.0424)	**.5147(.0295)**	.0053(.0008)	**.0042(.0004)**
32	50_20	.4378(.0476)	.4266(.0375)	.0046(.0006)	.0041(.0004)
33	50_20	.3422(.0266)	**.4383(.0838)**	.0125(.0006)	**.0069(.0051)**
34	50_20	.3828(.0384)	.4089(.0262)	.0036(.00005)	**.0030(.00029)**
35	50_20	.5558(.0349)	**.5763(.0165)**	.0025(.0005)	**.0020(.0001)**
36	100_20	**.3648(.0179)**	.2972(.0093)	**.010(.0003)**	.0130(.0006)
37	100_20	**.3442(.0142)**	.2844(.0101)	**.0086(.0003)**	.0107(.0003)
38	100_20	.3006(.0196)	.3025(.0136)	.0067(.0009)	.0066(.0002)
39	100_20	.6495(.0185)	.6515(.0191)	.0010(.0001)	.0010(.0001)
40	100_20	.3658(.0158)	**.3828(.0100)**	**.0085(.0003)**	.0088(.0002)

Table 5.1 reveals that GP-A-NSGA-III(PSO) performed significantly better than GP-NSGA-III in most of the test instances. In the case of HV, GP-A-NSGA-III(PSO) performed significantly better in 24 out of 40 test instances. On the other hand, GP-NSGA-III performed significantly bet-

ter only in 6 instances. For the remaining 10 instances, the two compared algorithms performed statistically the same. In regard to IGD, Table 5.1 exhibits the same pattern. GP-A-NSGA-III(PSO) achieved significantly better performance in 21 out of the 40 test instances. In contrast, GP-NSGA-III only performed significantly better in 13 instances.

Table 5.1 also shows that GP-A-NSGA-III(PSO) not only performed better on smaller instances compared to the baseline algorithms GP-NSGA-III. Also, it is more effective in more challenging and larger instances. For some test instances (e.g., instances 1, 10, 24), GP-A-NSGA-III(PSO) achieves a significant improvement. This demonstrates the usefulness of the proposed adaptive reference point scheme, which can find a better association with population members and obtain well-distributed reference points.

Further analysis

To further investigate how the adaptive reference point scheme affects the GP search process, we plot the average number of useless reference points (those associated with no individual in the population). Moreover, we plot the average HV and IGD of the non-dominated solutions obtained for each generation during the 30 independent runs of the two compared algorithms, as given in Figures 5.3 and 5.4, respectively.

Figure 5.4 shows the convergence curves of the HV and IGD values of the non-dominated solutions on the training set. Figure 5.4 shows that the adaptive reference point scheme can significantly reduce the number of useless points during the GP search process. Without adaptive points, the number of useless references points in GP-NSGA-III increased from 910 to about 980. On the contrary, in GP-A-NSGA-III(PSO), the number of useless reference points first increased and then decreased to 810. This figure reveals that the adaptive reference point scheme at the later stage of the search led to less useless reference points. Thus, it was a better refinement of the densely populated regions of the population. Figure

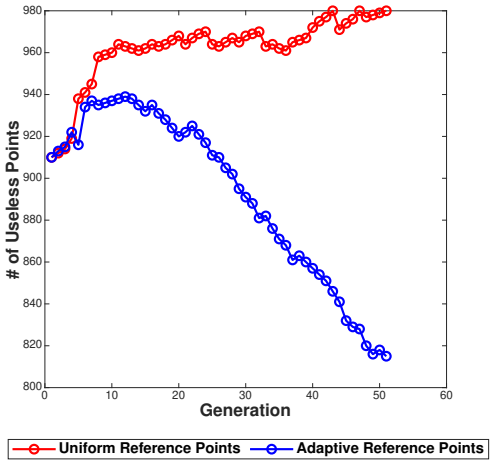


Figure 5.3: The curves of the average number of useless reference points in GP-NSGA-III, GP-A-NSGA-III(PSO).

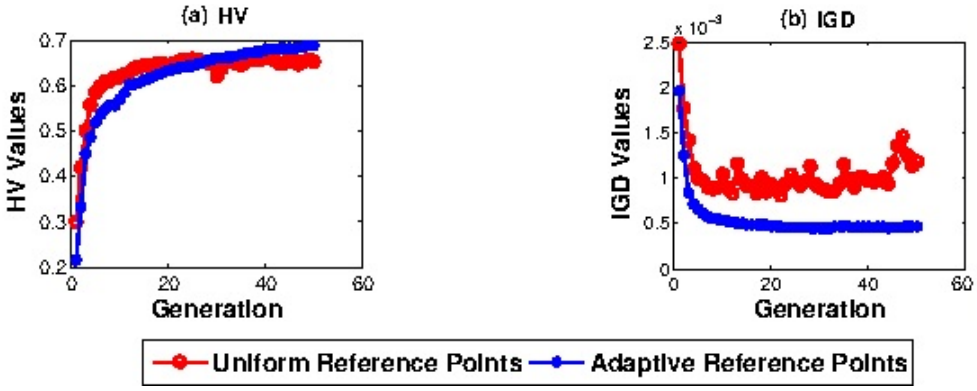
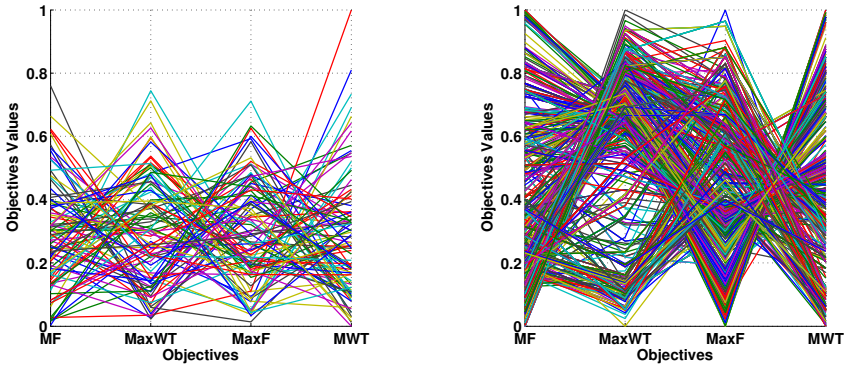


Figure 5.4: The curves of the HV and IGD values of the non-dominated solutions on the training set during the 30 independent GP runs.

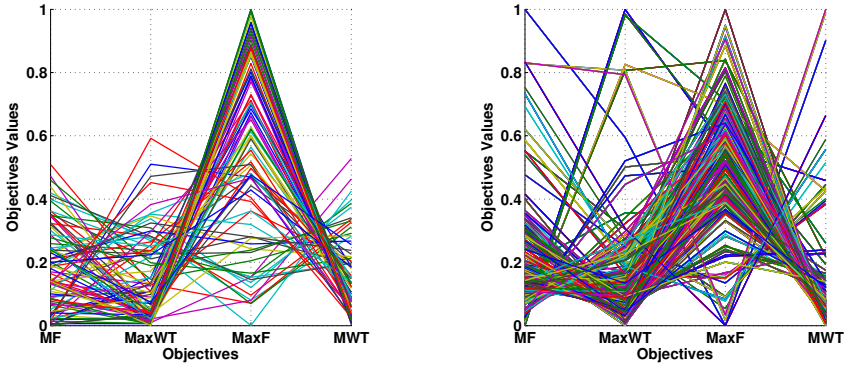
5.4 reveals that the GP-A-NSGA-III obtained better convergence curves in terms of both HV and IGD. This figure reveals that the reduction of useless reference points can lead to better non-dominated sets.

Figures 5.5 (a) to 5.6 (b) show the distribution of the reference points and the population's fitness values in generations 1 and 50 of GP-ANSGA-



(a) Distribution of the reference points. (b) Distribution of the fitness values.

Figure 5.5: Parallel coordinate plots of GP-A-NSGA-III(PSO) at generations 1.



(a) Distribution of the reference points. (b) Distribution of the fitness values.

Figure 5.6: Parallel coordinate plots of GP-A-NSGA-III(PSO) at generations 50.

III(PSO). It can be seen that in generation 1, the reference points are close to the initial uniform distribution, and the fitness distribution of the population is relatively uniform as well. On the other hand, at generation 50, the distributions of the reference points and the population's fitness val-

ues become very similar. This is consistent with our expectation which is to use a similar distribution of reference points as that of the population to fine-tune the promising area around the Pareto-front.

From this study, we identify a key research issue of having non-uniform Pareto-front, i.e., the simple adoption of uniformly distributed reference points failed to promote solution diversity during evolution and affected the performance of algorithms.

In conclusion, the proposed reference point adaptation mechanism has the potential to decrease useless reference points. It can significantly improve the performance of GP-HH and NSGA-III in terms of both HV and IGD. However, this research direction requires further investigation to reduce more useless reference points and improve the quality of evolved rules.

5.4 Model-based adaptive reference points generation

In the literature, NSGA-III has been extended to add and delete reference points in an adaptive manner [85, 86]. A-NSGA-III is an extended version of NSGA-III and has implicitly guided the distribution of solutions. A-NSGA-III first generates the uniformly distributed reference points then adds and removes these points in a high dimension space. However, in a high dimension space, the removal condition (ρ is less than one) is difficult to achieve. The algorithm keeps on including additional reference points that clearly affect the algorithm's performance. Most of the adaptive reference point approaches cannot explicitly construct the distribution model. The model's construction helps the concrete knowledge of the Pareto-front that can provide a close match between the reference points with the distribution of Pareto-optimal solutions.

Our model-based approach in this thesis adopts a modeling technique

[30] that learns the distribution of the Pareto-optimal solution and generates the reference points according to the solution distribution. This section introduces the model with and without the Gaussian process model [30, 170] (see in Subsection 2.1.8 of Chapter 2). The model without the Gaussian model is called a *density-based model* that estimates the density of solutions from each pre-defined sub-region. After the density-based model, we build a Gaussian process-based probabilistic model which can reduce the density noise and provides a reliable approximation of the true shape of the Pareto-front. In this manner, this section will show the model effectiveness of the density-based model and the Gaussian process model.

5.4.1 Reference point Adaptation by density-based model

Our proposed algorithm (GP-NSGA-III with density model-based reference point adaptation (GP-NSGA-III-DRA)) can be considered as a major enhancement of GP-NSGA-III. GP-NSGA-III-DRA starts after the population update mechanism of GP-NSGA-III which can be seen in Subsection 4.3.2 of Chapter 4. GP-NSGA-III-DRA utilized the P_{g+1} population from GP-NSGA-II and predefined simplex locations W.

Formation of density-based model

The density-based model estimates the density of solutions at each sub-location (a location on a normalized hyperplane) $w \in$ W. Building this density-based probabilistic model consists of two steps. First, the density-based model evenly decomposes the simplex into several sub-locations $w_1, w_2, w_3.., w_k \in W$. This decomposition uses Das and Dennis's [39] systematic approach (The detailed information can be seen in Subsection 2.1.6 of Chapter 2). Then the association between each solution s with w is obtained based on their perpendicular distance $(\perp)$. As a result, a solution is associated with a sub-location($\hat{w}$) where the perpendicular distance between the s with w has a minimum value $\hat{w}$. The solutions associated with

$\hat{w}$ is recorded in archive $E(\hat{w})$.

The number of associated solutions in any sub-location $\hat{w}$ is obtained by dividing the number of the associated solutions in $E(\hat{w})$ by the total number of non-dominated solutions ($\parallel S \parallel$). The density-based probabilistic model is therefore defined as

$$P(D|\hat{w} \in W) = \frac{\parallel (E(\hat{w})) \parallel}{\parallel S \parallel}, \tag{5.2}$$

The formation of the density model is shown in **Algorithm** 6. Figures 5.7 and 5.8 show an example to illustrate the above procedure of formation of density model. First, solutions are associated with their closest reference points, which is presented in Figure 5.7. For instance, three solutions are closest to w_3. Then the algorithm calculates the density of solutions at each sub-simplex location and builds a density-based model using equation (5.2). This can be seen in Figure 5.8.

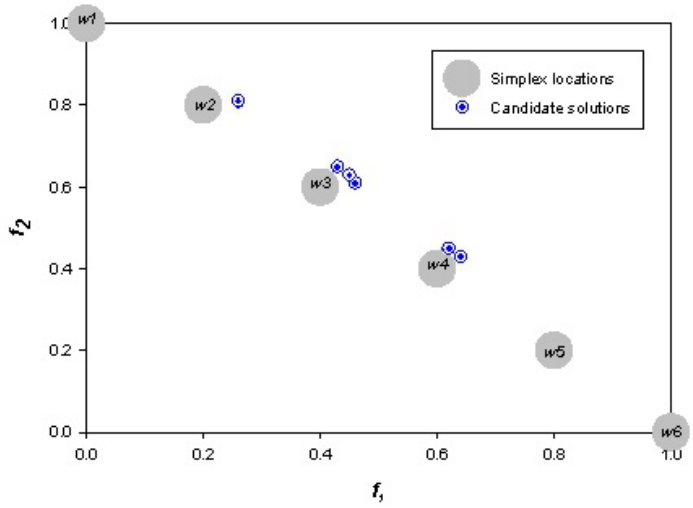


Figure 5.7: Solutions are closest to the reference points.

Algorithm 6: $Construct\_DensityModel(S_g, W)$

Input : S_g, W
Output: $\boldsymbol{Z_g^*}$

1 **foreach** $w \in W$ **do**
2 $E(w) = \emptyset$;
3 $D(w) = \emptyset$;
4 **end**
5 **foreach** $s \in S_g$ **do**
6 **foreach** $w \in W$ **do**
7 compute $d^{\perp}(s, w)$; `// perpendicular distance of each solution from` w
8 **end**
9 Assign $\hat{w} = argmin_{s \in S} d^{\perp}(s, w)$; `// associate the solution with the sub-location` w
10 Save s in $E(\hat{w})$;
11 **end**
12 **for** *i= 1 to* $\parallel E(\hat{w}) \parallel$ **do**
13 Assign $P(D|\hat{w}) = \parallel E(\hat{w})_i \parallel \div \parallel S \parallel$; `// probability of the associated solution`
14 Assign $D(\hat{w}) = \parallel P(\hat{w}) \parallel$*length of reference points; `// return solution's density`
15 **end**
16 $\boldsymbol{Z_g^*}$= Generate ($\boldsymbol{E}(\hat{w})$, $\boldsymbol{D}(\hat{w}), S_g, W$);
17 **return** $\boldsymbol{Z_g^*}$;

Reference points generation

Our proposed algorithm is broken into two sub-routines: (1) references points in proximity to the vertex and (2) the internal reference points. The generation of reference points is activated in line 16 of Algorithm 6.

1. **References points close to the vertex:** This method of the proposed algorithm handles the issue of A-NSGA-III [85] which relates to the generation of the reference points close to the vertices of the simplex. Our proposed algorithm generates reference points in close proximity to the simplex vertices. As a result, our algorithm enhances the

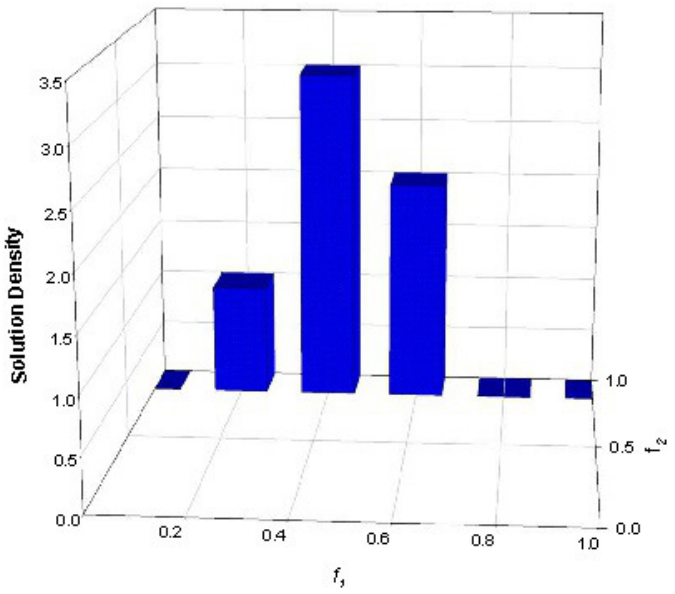


Figure 5.8: Density of solution at each sub-location of the simplex.

ability to generate reference points that match the distribution model closely. The detailed procedure is explained below:

(a) Obtain the center location (centroid) from the existing solutions in the sub-location $\hat{w} \in W$ where $\hat{w}$ is one of the locations close to the vertices of the hyperplane.

(b) Calculate the perpendicular distance from the centroid to associated solutions of $\hat{w}$.

(c) Select a solution s based on a minimum perpendicular distance.

(d) Calculate a mid-point value between the selected solution and the central location for generating a corresponding reference point. This mid-point of each dimension is considered as one of the reference points around the vertices.

(e) Selected solution s will not be the part of a centroid. Thus, the

Algorithm 7: $\boldsymbol{E}(\hat{w})$, $\boldsymbol{D}(\hat{w}), S_g, W$

Input : $\boldsymbol{E}(\hat{w})$, $\boldsymbol{D}(\hat{w}), S_g, W$
Output: $\boldsymbol{Z_g^*}$

foreach $\hat{w} \in W$ **do**
  set nref= $\| D(\hat{w}) \|$; `// number of reference points required at location` $\hat{w}$
  Assign $Z^r = \hat{w}$; `// set` $\hat{w}$ `as a first reference point`
  if *Z^r!=Vertex Points* **then**
    Assign $\boldsymbol{Z_g^*} = IntermediatePoints(E(\hat{w}), D(\hat{w}), nref, \overline{S}_g, W, Z^r)$; `// call intermediate points method`
  end
  if *Vertex points* **then**
    Assign $\boldsymbol{Z_g^*} = VertexPoints(E(\hat{w}), D(\hat{w}), nref, \overline{S}_g, W, Z^r)$; `// call vertex points method`
  end
end
return $\boldsymbol{Z_g^*}$;

next centroid is calculated from existing solutions that are still in the race of acquiring reference points.

(f) Repeat steps (a) to (e) until the required number of reference points are generated based on the size of the number of associated solutions with $\hat{w}$.

2. **Generation of reference points internal to the simplex:** An example below shows the reference points generation internal to the simplex. Consider the j-th internal simplex location which is associated with more than one solution. If the situation has $M = 3$ objectives, then M points are generated at the vertices of j-th internal locations on the simplex. The simplex side length (interval) is equal to the distance between two neighboring internal locations on the originally specified hyperplane and store in the archive Z_p. This example is shown in Figure 5.9. In this example, $Z^1, Z^2, Z^3 \in Z_{new}$ reference points are generated around the j-th internal locations ($Z_j \in Z^r$). The

reference points generated by the following two equations:

$$points^i = Z_j - Z_p, \tag{5.3}$$

$$Z^i_{new} = points^i + (Interval)/M. \tag{5.4}$$

where the interval is the difference between two consecutive reference points on the hyperplane.

The above two equations kept the j-th internal locations at the center of the newly generated reference point. These newly generated reference points can be inserted in the reference points archive called Z_{new} if they satisfy the two main conditions: (i) a reference point must be inside the entire simplex boundary; (ii) every reference point must occupy a unique location on the simplex. Once new reference points are added into archive Z_{new}, then the association between existing members of Z_{new} and solutions in $E(\hat{w})$ must be checked. If the i-th reference point from Z_{new} still has $\rho_i \geq 2$, again new reference points are generated around the i-th reference points, but this time, the interval's parameter value is halved. Therefore, new reference points are getting closer to the associated solutions with the original reference points and increase the chance of getting an ideal association (ρ =1). This process is also shown in Figure 5.9. Figure 5.9 demonstrates that the ith reference point is kept as a centroid location for newly generated reference points and the reference points are layered. These layers are also shown in Figure 5.9 with two different colors (blue and black lines). Thus, we named this method a layered centroid approach.

5.5 Gaussian process-based probabilistic model

In Algorithm 6, we defined the density-based model which is built to the model on the solution's density at each location of the simplex. This sec-

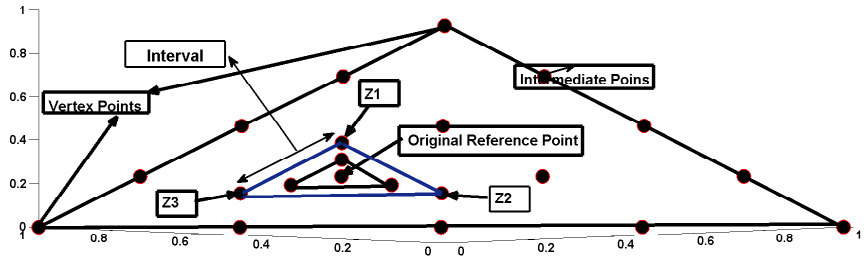


Figure 5.9: Generate reference points until $M-1$ times

tion predicts a mean value in sub-location for generating reference points and subsequently reducing the density noise. Gaussian processes (GaP) (The detailed information can be seen in Subsection 2.1.8 of Chapter 2) as a powerful method to model unknown functions. In particular, in this research objective, we used GaP-based modeling. The GaP is defined as a mean and a covariance function.

$$f(x) \sim GaP(\mu(x), K(x_i, x_j)), \tag{5.5}$$

K defines the shape of GaP and the choice of an appropriate kernel. The kernels' selection is based on assumptions such as patterns to be expected in the data. However, in this study, our assumption provides smoothness to the model which can reduce the noise of the model. Next, we calculate the area under the mean function. This area under the curve helps us to find the required number of reference points in each sub-location.

In our proposed algorithm, model-based adaptive reference points (GP-MARP-NSGA-III), we use the squared exponential kernel [170]. In this kernel, if x_i and x_j behave similarly then the function values at these points, $f(x_i)$ and $f(x_j)$, can be expected to be similar. The squared exponential kernel is also known as the Gaussian kernel.

$$K\left(x_i, x_j\right) = \sigma_f^2 \exp\left(-\frac{1}{2l}\left|x_i - x_j\right|^2\right). \tag{5.6}$$

where σ_f^2 and l are hyperparameters of the covariance functions.

5.5.1 Modelling by Gaussian process

In GP-MARP-NSGA-III, first, we construct the density model. The details of the construction of the density model of the simplex are described in Subsection 5.4.1. However, we construct the model by using mid-point location(w_c) of the two neighborhoods $\hat{w}_i \in W$ and $\hat{w}_j \in W$ on simplex. The reason for selecting mid-point locations because, in later this algorithm, we approximate areas under the mean function by using Riemann sum with the midpoints [48].

For the model construction, we need to find two sub-locations $\hat{w}_i$ and $\hat{w}_i$ those are close in perpendicular distance with each other. These neighboring locations of $\hat{w}_i$ are further saved in an archive $Negh(w_i)$. Next, the mid-point between $\hat{w}_i$ and $\hat{w}_j$ is determined and stored in W_c. This center location at $w_c \in W_c$ serves as an input of our density-based model. Then, the model based on the association between the center locations and their respective solutions is stored in $E(\hat{w}_c)$. The detailed information can be seen in Algorithm 8.

Specifically, the total number of the associated solutions with $\hat{w}_c$ ($\hat{w}_c$ $\hat{w}_c$ means minimum perpendicular distance with population members) is recorded in archive $D(\hat{w}_c)$. In accordance with $D(\hat{w}_c)$, the solution density of each sub-location $\hat{w}_c$ is calculated by the associated solutions in $E(\hat{w}_c)$ divided by the total number of non-dominated solutions ($\parallel S \parallel$) of each generation. The density-based probabilistic model($P(D|\hat{w}_c \in W_c)$) is defined in Equation (5.2).

The GaP is a sample of a stochastic process. In this way, $P(D|\hat{w}_c \in W_c)$ can be seen as a latent function [170] where the joint distribution of a infinite number of these variables $P(D(w_{c1}), \ldots, D(w_{ck}))$ is itself Gaussian:

$$P(D|\hat{w}_c \in W_c) = N(D|\mu, K), \tag{5.7}$$

After the construction of the density model, we train the GaP means to estimate the conditional probability of $P(D|\hat{w} \in W)$ which is fully specified by the mean function $\mu(w_i)$ and K that calculates the covariance be-

tween any two sub-locations(w_{ci}, w_{cj}). Without the loss of generality, we set zero to the prior mean function common in practice.

$$P(D|\hat{w_c} \in W_c) = N(D|0, K), \tag{5.8}$$

With equation (5.8), we make predictions D_* of given inputs W_{c*} from the posterior distribution $P(D_*|\hat{w_c} \in W_c, D, \hat{w_{c_*}} \in W_{c_*})$ .The posterior distribution can be obtained as a Gaussian distribution with mean and variance.

$$P(D_*|\hat{w_c} \in W_c, D, \hat{w_{c_*}} \in W_{c_*})) = N(D_*|\mu_*, K_*). \tag{5.9}$$

The detailed information of the GaP can be seen from the Subsection 2.1.8 of Chapter 2. The detailed process of GaP modeling is shown in Algorithm 8 and in Figure 5.10. In this Figure, we show Gaussian process model and get a smooth function by reducing model-noise. This function helps us to calculate the area under a mean function of the GaP.

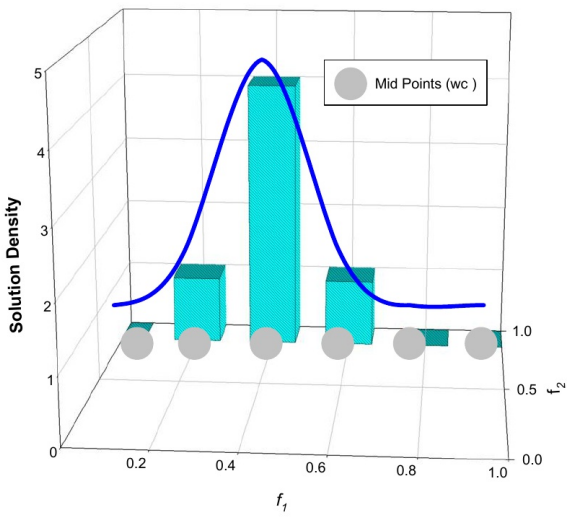


Figure 5.10: Train a Gaussian process on density-based model.

Algorithm 8: $\boldsymbol{Gaussion\_Modelling(S_g, W)}$

Input : S_g, W

Output: $\boldsymbol{E(\hat{w_c})}$ (at $\hat{w_c}$) &$\boldsymbol{\mu(\hat{w_c})}$ (mean prediction at $\hat{w_c}$)

foreach $w_i \in W$ **do**
  foreach $w_j \in W$ **do**
    compute $d^{\perp}(w_i, w_j)$; // find neighbors of w_i
  end
  Assign $\hat{w}_{neg} = argmin_{w_j \in W} d^{\perp}(w_i, w_j)$ Save $\hat{w}_{neg}$ in $Neg(w_i)$; ; // save neighbors of w_i location in $Neg(w_{ci})$
end
foreach $w_{neg} \in Neg(w_i)$ **do**
  Calculate the center location w_c between $w_{neg} \in Neg(w_i)$ and w_i ; // find center loction between w_i and w_{neg} location in $Neg(w_i)$
  Save w_{ci} in $E(w_{ci})$; // save center location in $E(w_{ci})$
end
foreach $s \in \overline{S}_g$ **do**
  foreach w_{ci} *in* $E(w_{ci})$ **do**
    compute $d^{\perp}(s, w_{ci})$
  end
  Assign $\hat{w_c} = argmin_{s \in S} d^{\perp}(s, w_{ci})$; // associate the solution with the center location
  Save s in $A(\hat{w_{ci}})$
end
foreach $s \in A(\hat{w_{ci}})$ **do**
  Calculate the number of associated solutions with $\hat{w_{ci}}$ and store in $D(\hat{w_{ci}})$
end
Assign $P(D|\hat{w_{ci}}) = \| A(\hat{w_{ci}}) \| \div \| S \|$; // build a density model
Train a Gaussian process on density-based model.$P(D|\hat{w_{ci}}) \sim N(0, K)$
Test input $\hat{w_c}*$ and obtain the Gaussian distribution with mean $\mu(\hat{w_{c*}})$ **return** $\boldsymbol{E}(\hat{w_c})$ &$\boldsymbol{\mu}(\hat{w_c})$

5.5.2 Calculate the area under the mean function

In this method, we approximately calculate the area under the mean-function through the mid-point rule. This area under the mean-function helps us to find the required number of the reference points in each sub-

Algorithm 9: $\boldsymbol{Calculate\_Area(\mu(\hat{w}), , E(\hat{w}_c))}$

Input : $\mu(\hat{w}_c), E(\hat{w}_c)$
Output: $Z(\hat{w}_c)$
foreach $\hat{w}_c \in E(\hat{w}_c)$ **do**
  Assign $area = \frac{b-a}{n} * (\mu(w_c))$
  Save area in $Ar(w_c)$
end
Total Area: $\int_a^b \mu(\hat{w}_c)dx = \frac{b-a}{n}(\mu(w_{c1}) + \mu(w_{c2}) + \mu(w_{c3}) + \cdots + \mu(w_n)) = Ar(w_{c1}) + Ar(w_{c2}) + \cdots + Ar(w_{cn}))$
foreach $\hat{w}_c \in Ar(\hat{w}_c)$ **do**
  Calculate the number of reference points ($nref$) in each $\hat{w}_c$: $\frac{Ar(w_c)}{TotalArea}$;
  Save $nref$ in $Z(\hat{w}_c)$; `// save number of reference points in` $Z(\hat{w}_c)$
end
return $\boldsymbol{Z}(\hat{w}_c)$;

location. We decided to use the mid-point rule for calculating the area under the mean function. This is defined in equation (5.10)

$$TotalArea \approx \int_a^b \mu(\hat{w}_c)dx = \frac{b-a}{n}(\mu(w_{c1}) + \mu(w_{c2}) + \cdots + \mu(w_n)), \quad (5.10)$$

Next we have found the number of reference points in each specified area and they can be determined as below

$$referencepoints = \frac{Ar(w_c)}{TotalArea}. \quad (5.11)$$

where $Ar(w_c)$ is a specific area under the curve. The detailed process for area calculation is further presented in Algorithm **9**. This can also be seen in Figure 5.11. Next, the algorithm estimates the number of reference points at each sub-location w_c using equation (5.11) and generates the required reference points in each sub-area.

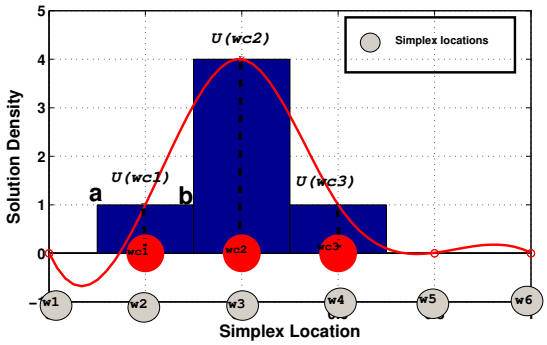


Figure 5.11: Area under the mean function of the Gaussian process model

5.5.3 Reference points generation

The generation of reference points is described in this section. For each $\hat{w}_c$ in the reference points archive, if the number of reference points, i.e., $Z(\hat{w}_c)$ has a niche count equal to one, there is no need to generate new reference points. Otherwise, reference points are generated using the equations (5.3) and (5.4). The rest of the procedure for the newly generated reference points is similar to the "Generation of reference points internal to the simplex" of Subsection 5.4.1.

5.5.4 Computational Complexity of One Generation of GP-MARP-NSGA-III

In line 3 of Algorithm 8, we are computing a neighboring location of W, where its size is equal to the size of population N. So, computing a neighboring location takes $\mathcal{O}(MN^2)$ where M is the objective number. After that, the center location of two neighboring locations of W (line 8 of Algorithm 8) requires $\mathcal{O}(MN)$. In line 13 of Algorithm 8, associating a population of N individuals to N reference directions takes $\mathcal{O}(MN^2)$. However, to build a density model (line 21) requires only $\mathcal{O}(N)$. In addition, to train a GaP on the density-based model (line 22) $\mathcal{O}(MN^3)$, which totally depends on the kernel (K_{NxN}). Finally, the computational complexity of obtaining

mean μ (line 23) is $\mathcal{O}(N)$.

All operations in Algorithm 9 (lines 2, 5, and 8) in calculating area and calculating the number of reference points would require $\mathcal{O}(N)$ computations. Thereafter, the reference points generation (the procedure shows in Algorithm 7) requires $\mathcal{O}(MN)$ comparisons.

Considering all the above considerations and computations, the overall worst-case complexity of one generation of GP-MARP-NSGA-III is $\mathcal{O}(MN^3)$.

5.5.5 Design of experiment

To verify the effectiveness of the proposed model-driven reference points, we compare the performance of our algorithm GP-MARP-NSGA-III and GP-NSGA-III-DRA with the GP-NSGA-III, GP-A-NSGA-III in the experimental studies. We selected the Taillard (TA) static JSS benchmark instances [182] as the testbed. The details of TA can be found in Section 3.1 of Chapter 3. In the experiments, we considered four potentially conflicting objectives:the mean flowtime (mF) (see equation (2.1) of Chapter 2) , maximal flowtime (maxF) (see equation (2.2) of Chapter 2), mean weighted tardiness (mWT) (see equation (2.6) of Chapter 2) and maximal weighted tardiness (maxWT) (see equation (2.7) of Chapter 2). For all the compared algorithms, 30 independent runs were conducted.

All the compared algorithms adopt the tree-based representation of dispatching rules. The terminal set and function set are described in Section 3.2 of Chapter 3. GP-MARP-NSGA-III, GP-NSGA-III-DRA and GP-A-NSGA-III adopt the same parameter setting of the baseline algorithm GP-NSGA-III. In the experiments, the two commonly used measures, i.e. IGD [206] and HV [212] are used to compare the algorithms (see in Section 3.3 of Chapter 3).

Table 5.2: The mean and standard deviation of HV and IGD values of the 30 independent runs on training instances of the compared algorithms on four-objective JSS problems.

HV				
Statistic	GP-NSGA-III	GP-A-NSGA-III	GP-NSGAIII-DRA	GP-MARP-NSGA-III
$(\bar{x} \pm \sigma)$	0.67234(0.02245)	0.67561(0.02043)	0.67655(0.01955)	**0.68498(0.01961)**
IGD				
Statistic	GP-NSGA-III	GP-A-NSGA-III	GP-NSGA-III-DRA	GP-MARP-NSGA-III
$(\bar{x} \pm \sigma)$	0.00139(0.00012)	0.00131(0.00012)	0.00130(0.00012))	**0.00127(0.00008)**

5.5.6 Results and discussion

For each algorithm, 30 GP runs have been performed to obtain 30 final sets of dispatching rules. Afterward, the rules are tested on the 40 test instances. Tables 5.2 and 5.3 show the mean and standard deviation of the average HV and average IGD values obtained by GP-NSGA-III, GP-A-NSGA-III, GP-NSGA-III-DRA, and GP-MARP-NSGA-III. The Wilcoxon rank-sum test [195] with the significance level of 0.05 has been applied separately to compare both the HV and IGD achieved by all algorithms. The significantly better results are bolded.

Table 5.3: The mean and standard deviation of HV and IGD achieved by all competing algorithms on test instances on four-objective JSS problems.

HV				
Statistic	GP-NSGA-III	GP-A-NSGA-III	GP-NSGAIII-DRA	GP-MARP-NSGA-III
$(\bar{x} \pm \sigma)$	0.47422(0.024231)	0.48896(0.02336)	0.48625(0.02445)	**0.49844(0.02147)**
IGD				
Statistic	GP-NSGA-III	GP-A-NSGA-III	GP-NSGA-III-DRA	GP-MARP-NSGA-III
$(\bar{x} \pm \sigma)$	0.00177(0.00020)	0.00162(0.00025)	0.00164(0.00027)	0.00161(0.00016)

Table 5.2 reveals that GP-MARP-NSGA-III achieved significantly better performance in HV than the other algorithms. In terms of IGD, GP-MARP-NSGA-III clearly outperformed GP-A-NSGA-III, GP-NSGA-III-DRA, and GP-NSGA-III. Moreover, our proposed algorithm GP-NSGA-III-DRA per-

formed significantly better than GP-A-NSGA-III and GP-NSGA-III.

Table 5.3 summarizes the testing performance of all algorithms in terms of IGD and HV. The obtained test results exhibit the same patterns as the training performance results. In the case of HV, GP-MARP-NSGA-III performed significantly better than the other compared algorithms. In the case of IGD, GP-MARP-NSGA-III is competitive with GP-A-NSGA-III. However, both our proposed model-driven algorithms performance is still significantly better in IGD than the GP-A-NSGA-III and GP-NSGA-III.

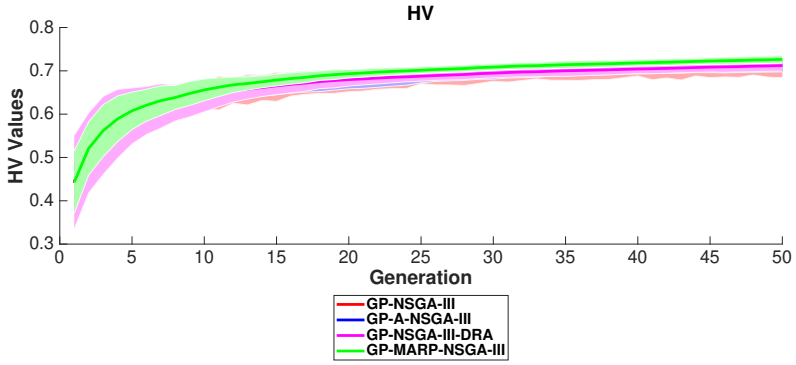


Figure 5.12: HV values of the non-dominated solutions on the training set during the 30 independent GP runs.

Further analysis

For further analysis of the algorithms' performance, we plotted the average HV and IGD of the non-dominated solutions obtained so far for each generation during the 30 independent runs of the four compared algorithms.

Figures 5.12 and 5.13 show convergence curves on the training sets. Figures 5.12 and 5.13 reveal that during the first few generations of evolution, all algorithms exhibited similar HV and IGD values. However, GP-MARP-NSGA-III starts to outperform other competing algorithms. Moreover, when the solutions are very close to the Pareto-front, GP-MARP-

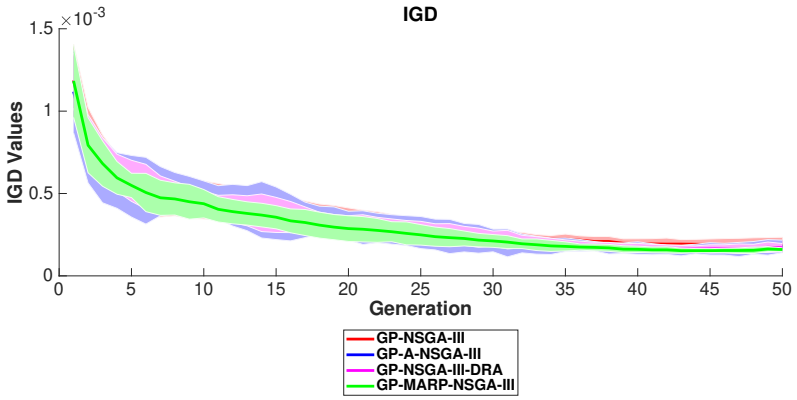


Figure 5.13: IGD values of the non-dominated solutions on the training set during the 30 independent GP runs.

NSGA-III achieved significantly better HV and IGD. This result clearly demonstrates that GP-MARP-NSGA-III can converge to a high-quality Pareto-front.

A useless reference points plot can help us to understand how well the reference points are generated from model-based approaches that match the distribution of solutions. From Figure 5.14 shows that the number of useless points during the GP search process is constantly decreasing in both model-based reference points adaptation methods. Figure 5.14 shows that GP-MARP-NSGA-III has less useless reference points than GP-NSGA-III-DRA. This reveals that reducing the density noise led to having fewer useless reference points and enhanced the better matches between the reference points and candidate solutions. Further, the evolved Pareto-front can help enhance solution diversity and, therefore, the algorithms' performance.

The parallel coordinate plots for all competing algorithms are shown in Figures 5.15 (a) to 5.18 (b). It is well known that the number of useless reference points will increase if the distribution of reference points is not fully matched with the distribution of the candidate solutions. This can be seen in Figures 5.15 (a) and 5.15 (b) for GP-NSGA-III. In these figures,

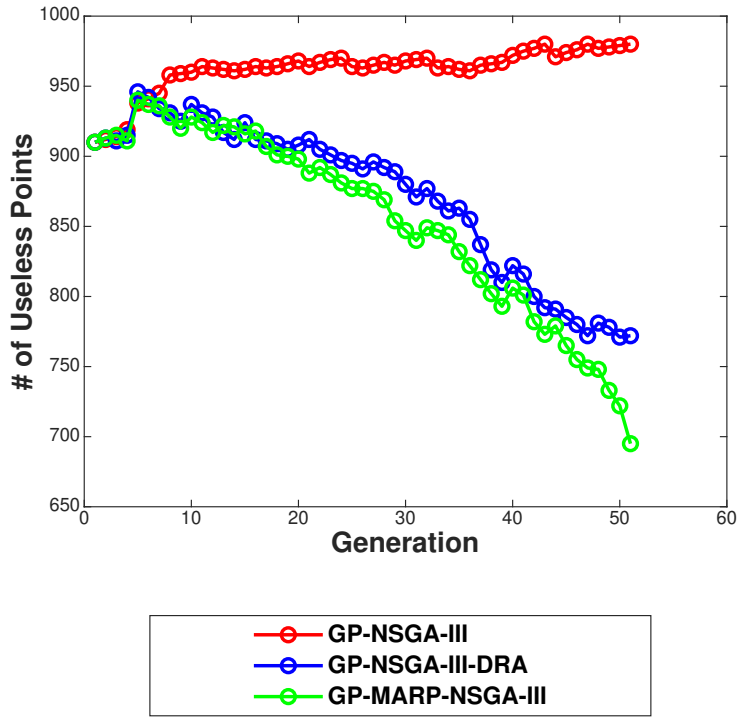
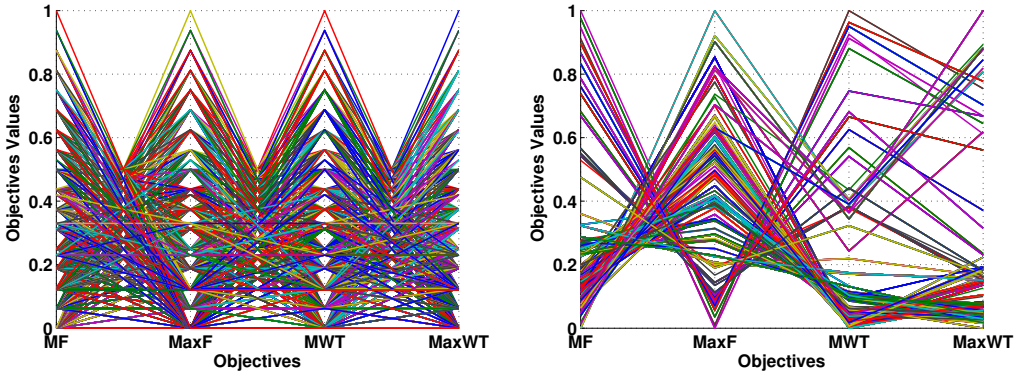


Figure 5.14: The curves of the average number of useless reference points in GP-NSGA-III, GP-NSGA-III-DRA, and GP-MARP-NSGA-III.

we can see that the distribution of the reference points is uniform and that some reference points are far away from the solution locations, especially on MWT and MaxWT objectives. These useless reference points will cause a negative impact on algorithm performance, as confirmed by experiment results reported in Table 5.2.

Figure 5.16(a) shows that GP-A-NSGA-III still has a few useless points. Figures 5.16 (a) and 5.16 (b) show that solutions are also not well associated with reference points on MAxF and MaxWT objectives. In contrast, Figures 5.18 (a), and 5.18 (b) show that the reference points' distributions and the population closely match each other in the GP-MARP-NSGA-III

which is better than GP-NSGAIII-DRA. The distributions of the reference points and the population of GP-NSGA-III-DRA shown in Figures 5.17 (a) and 5.17 (b).



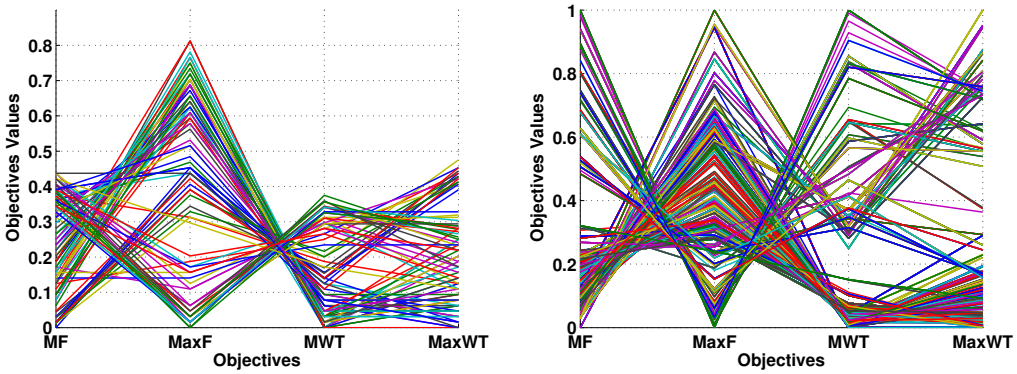
(a) Distribution of the reference points. (b) Distribution of the fitness values.

Figure 5.15: Parallel coordinate plot for the distribution of the reference points and the distribution of the fitness values at generations 50 of GP-NSGA-III.

5.6 Selection of adaptive reference points approach

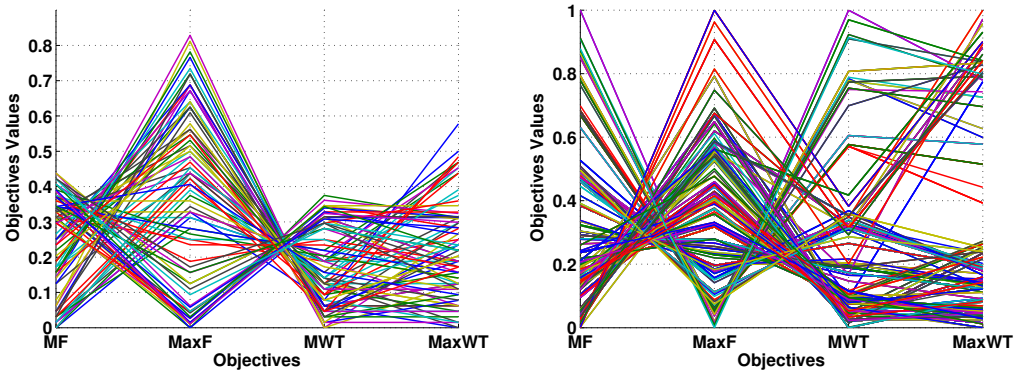
In this chapter, we proposed three adaptive reference points approaches. Each reference point approach is equally important but the selection of approach depends on the decision-makers specific requirements as follow:

1. If the decision-maker prefers a very simple and efficient approach with reasonable effectiveness, the PSO-based adaptive reference points approach is the right choice for generating reference points adaptively.



(a) Distribution of the reference points. (b) Distribution of the fitness values.

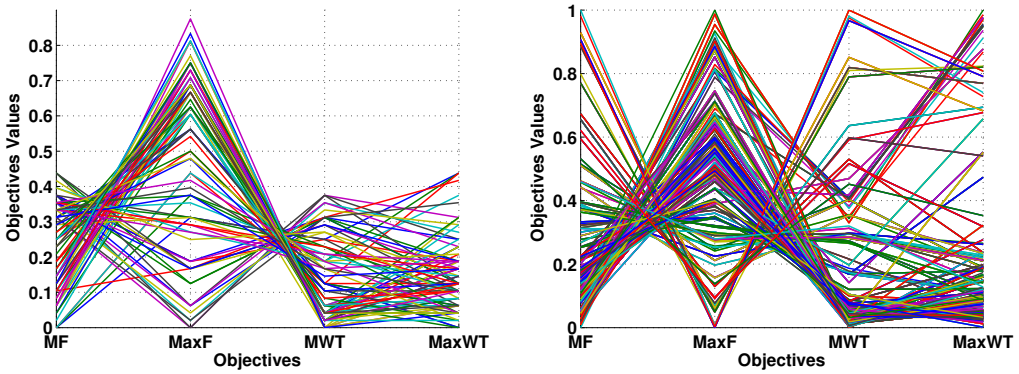
Figure 5.16: Parallel coordinate plot for the distribution of the reference points and the distribution of the fitness values at generations 50 of GP-A-NSGA-III.



(a) Distribution of the reference points. (b) Distribution of the fitness values.

Figure 5.17: Parallel coordinate plot for the distribution of the reference points and the distribution of the fitness values at generations 50 of GP-NSGA-III-DRA.

2. If the decision-maker prefers an effective approach other than the



(a) Distribution of the reference points. (b) Distribution of the fitness values.

Figure 5.18: Parallel coordinate plot for the distribution of the reference points and the distribution of the fitness values at generations 50 of GP-MARP-NSGA-III.

PSO-based approach, the density-based model adaptive reference points approach is a good choice for generating reference points adaptively. Further, the density model is more efficient than our GaP model because it does not need to maintain the Gaussian matrix. Also, the density model is more flexible than the GaP model because it introduces simple sampling techniques. The density-based model is a better trade-off approach between the GaP model approach and the PSO-based approach.

3. If the decision-maker prefers an effective approach which approximates the Pareto-front more accurately than the PSO-based approach and density-based model approaches, the GaP model is the right choice for the reference points generation. However, the GaP model approach compromises simplicity, efficiency and flexibility as compared to the other two approaches.

5.7 Chapter summary

This chapter aims to identify key research issues of those algorithms that use uniformly distributed reference points, including NSGA-III. The simple adoption of uniformly distributed reference points failed to promote solution diversity during evolution and affected the performance of the many-objective optimization problems, which irregular, disconnected, degenerate, and inverted shapes of Pareto-front.

To achieve this goal, we proposed model-free and model-driven adaptive generation mechanisms for reference points. In our model-free mechanism, we introduced a new reference point adaptation mechanism inspired by PSO. Essential changes to particle dynamics in PSO have also been introduced in our mechanism to prevent the majority of reference points from converging to small areas in the objective space. This algorithm outperformed the baseline algorithm GP-NSGA-III.

Model-based techniques explicitly construct the distribution model and give the learning ability to the algorithms. These algorithms have been designed to approximate the Pareto-front accurately and generate reference points that closely match the distribution of Pareto-optimal solutions. Additionally, improvements have also been made to effectively reduce the modeling noise in our density-based model using GaP modeling. The GP-MARP-NSGA-III algorithm defines the area under the mean-locations and generates the required number of reference points in each sub-area.

The proposed algorithm was applied to the static JSS problem. We compared our proposed algorithms with NSGA-III and previously proposed reference point adaptive approaches. Experimental results on the benchmark JSS problems show that our proposed algorithm reduces the useless reference points and provides a better distribution of Pareto-optimal solutions on the entire Pareto-front. Further, a better distribution of reference points also helps to improve the diversity of the solutions that

can be observed visually and in terms of HV and IGD.

The experimental results have demonstrated that the GP-MARP-NSGA-III performs significantly better problems with irregular Pareto-front because it reduces the number of useless reference points and provides well-diversified Pareto-optimal solutions on the entire Pareto-front. Further, a better distribution of reference points also helps to improve the diversity of the solutions that can be observed visually as well as in terms of HV and IGD. This finding leads us to believe that our algorithm performance is promising on non-uniformly and irregularly distributed Pareto-front. Also, in the future, these adaptive reference point methods might be easily deployed with any other reference points-based EMO algorithms to improve their performance on problems with irregular Pareto-front.

In the next chapter, we will enhance the exploitation ability of the GP-NSGA-III. This is realized by combining GP-HH with Pareto local search.

Chapter 6

GP with Pareto Local Search for Many-Objective JSS

6.1 Introduction

Genetic programming (GP) is considered the most popular method for discovering and constructing dispatching rules for scheduling problems [141, 149]. Previous studies have shown that GP has been successfully used to evolve very effective dispatching rules for job shop scheduling (JSS) problems automatically [159, 143].

Researchers have studied the application of Pareto local search (PLS) to multi-objective evolutionary algorithms (MOEAs) with some success [50, 26]. By hybridizing global (such as genetic algorithms (GA)) search with local search, the performance of many evolutionary algorithms (EAs) can be improved because the local search can help to enhance the exploitation ability of EAs [26, 79]. This hybridization motivates us to integrate GP as a primary global search method with PLS and improve the quality of the evolved rules for many-objective JSS.

PLS can be considered a direct extension of local search from single-objective problems to the multi-objective domain [26, 50]. PLS has three main algorithmic components [137]: (1) selection of solutions for neigh-

borhood exploration, (2) exploring the neighbors of the selected solutions from (1), and (3) describe the conditions under which a new solution stores in the PLS archive. The goal of PLS is to obtain a good approximation of the Pareto-optimal set. In particular, the study in [79] showed that suitable candidates for local search should be carefully selected based on a weighted sum of multiple objectives as a fitness function. One promising approach for improving the searchability of evolutionary multi-objective optimization (EMO) algorithms to find near Pareto-optimal solutions is the hybridization with local search [26, 79]. The hybridization with local search algorithms is often referred to as memetic algorithms [79]. Examples of such a hybrid algorithm can be found in multi-objective traveling salesman problems and flowshop problems [14, 137].

PLS is very effective for tackling *NP-hard* multi-objective JSS problems [14]. However, no research works have been dedicated to studying PLS in genetic programming based hyper-heuristic (GP-HH) for many-objective JSS. Based on our survey, there is only one existing work [149] studied using of local search techniques in GP-HH for single-objective JSS. Due to this limitation, we investigate the effectiveness of PLS in GP-HH in this chapter. This study is expected to inspire many future studies on PLS in GP-HH for many-objective optimization. Based on the investigation in this chapter, new GP-PLS algorithms will be developed. While developing our algorithm, we aimed to address three challenges that are vital for the seamless integration of PLS with GP-HH. First, the selection of initial solutions for neighborhood exploration. Second, a tree's neighborhood structure (dispatching rule) in GP is not defined in the literature. Third, the acceptance criteria during the local search have to be carefully designed to guide the search properly for trade-offs solutions in many-objective JSS problems.

A new fitness-based selection mechanism is proposed to address the first challenge, which uses a decomposition-based approach. To address the second challenge, a restricted mutation, is introduced to modify dis-

patching rules evolved by GP. The restricted mutation tries to avoid large changes, making the neighboring rules too different from their parent rule. Multiple consecutive local search steps will be performed to encourage the discovery of better rules surrounding an existing one. Meanwhile, the third challenge is tackled by adopting a dominance-based [26] and fitness-guided [79] acceptance strategy. These three challenges will be explained more in Section 6.2. In this chapter, we first perform the empirical study to determine whether the inclusion of local search improves the algorithms' performance. Then we propose the two new memetic algorithms that integrate GP with PLS. The chapter aims at developing GP-PLS and enhance the quality of evolved dispatching rules for many-objective JSS. The sub-objectives of this chapter are:

1. To investigate whether the inclusion of PLS within a GP-HH algorithm can increase the chance of discovering highly effective dispatching rules for many-objective JSS,

2. To develop a new fitness-based selection mechanism and neighborhood structure in GP-PLS, and

3. To compare the proposed GP-PLS with the state-of-the-art GP-NSGA-III algorithm on a group of benchmark JSS problems.

In this chapter, we start with an explanation of our proposed algorithm, GP-PLS-I which, uses a restricted neighborhood structure and the partial acceptance mechanism for many-objective optimization problems (MaOPs). This is followed with the explanation of our other proposed algorithm, GP-PLS-II, which is the extension of GP-PLS-I. Following this, we will expand on the overview of proposed algorithms, GP-PLS-I and GP-PLS-II. After the overviews of proposed algorithms, the experimental design, the results, the analysis, and the discussion for the efficacy investigation are covered. Finally, a chapter summary that wraps up this chapter is provided.

6.2 GP-PLS structure

This section describes the general framework of the proposed algorithms which combine GP with PLS. First, we propose GP-PLS-I, which combine PLS with GP. Then, we propose an extension of GP-PLS-I, GP-PLS-II. .

6.2.1 General framework of GP-PLS

GP-PLS starts with the initialization by using the ramped-half-and-half method. The quality of dispatching rules evaluate in terms of each objective (lines 1 and 19 of Algorithm 10). These rules then apply to a set of JSS training instances $\boldsymbol{I}_{train}$ to generate schedules for them. Then, for each objective, the quality of a rule p is defined as the average objective value of the schedules generated across all training instances.

Next, PLS features the use of an archive to keep track of candidate rules for local search. The archive initially has either a randomly-selected subset of rules (P_k) in the population (used in GP-PLS-I) or a selection of the subset of rules (P_k) based on their fitness value (used in GP-PLS-II). In both PLS algorithms, first, PLS selects K individuals from the population to form the archive. Then, the proposed algorithms iteratively search the neighboring solutions of every rule (p) in the archive with the mutation operator's help. A maximum of $step_{max}$ neighbors can be generated and the best neighbors (p_{new}) are compared with p. If the neighbor rule is better than p, then it is added into $\boldsymbol{P_{best}}$. This $(\boldsymbol{P_{best}})_{\boldsymbol{g}}$ archive represents the best-performing dispatching rules evolved so far.

6.2.2 GP-PLS-I overview

Algorithm 10 outlines the framework of GP-PLS-I. The algorithm of GP-PLS-I has three significant components, which can be seen in Figure 6.1. These components are 1) initialization and evaluation of dispatching rules, 2) PLS, and 3) NSGA-III selection. The initialization and evaluation of GP

Algorithm 10: The framework of GP-PLS.

Input : training set I_{train}

Output: A set of non-dominated solutions(rules) $\boldsymbol{P^*}$

Initialize of rules and Evaluate the population $\boldsymbol{P_0}$;
$g \leftarrow 0$;
while $g < g_{\max}$ **do**
    $\boldsymbol{P_{best}} \leftarrow \emptyset$;
    Randomly select K individuals from $\boldsymbol{P_g}$ to form $archive$;
    foreach $p \in archive$ **do**
        $p_{new} \leftarrow p$;
        for $step = 1 \rightarrow step_{max}$ **do**
            $p' \leftarrow$ mutate(p); `// neighbors`
            evaluate(p');
            if p' *is better than* p_{new} **then** $p_{new} \leftarrow p'$;
        end
        if p_{new} *is better than* p **then**
            $P_{best} \setminus p$;
            $(\boldsymbol{P_{best}})_{\boldsymbol{g}} \leftarrow (\boldsymbol{P_{best}})_{\boldsymbol{g}} \cup p_{new}$
        end
    end
    Apply genetic operators to $(\boldsymbol{P_{best}})_{\boldsymbol{g}} \cup (P_g \setminus (P_{best})_g)$ to generate offspring $\boldsymbol{Q_g}$;
    foreach $Q \in \boldsymbol{Q}_g$ **do** Evaluate rule Q;
    $g \leftarrow g + 1$;
end
return *The non-dominated individuals* $\boldsymbol{P^*} \subseteq \boldsymbol{P_{g_{\max}}}$;

are similar to the general framework of GP-NSGA-III (the detailed information can be seen in Subsection 4.3.2 of Chapter 4). PLS components are the same as the algorithmic components that have been described in

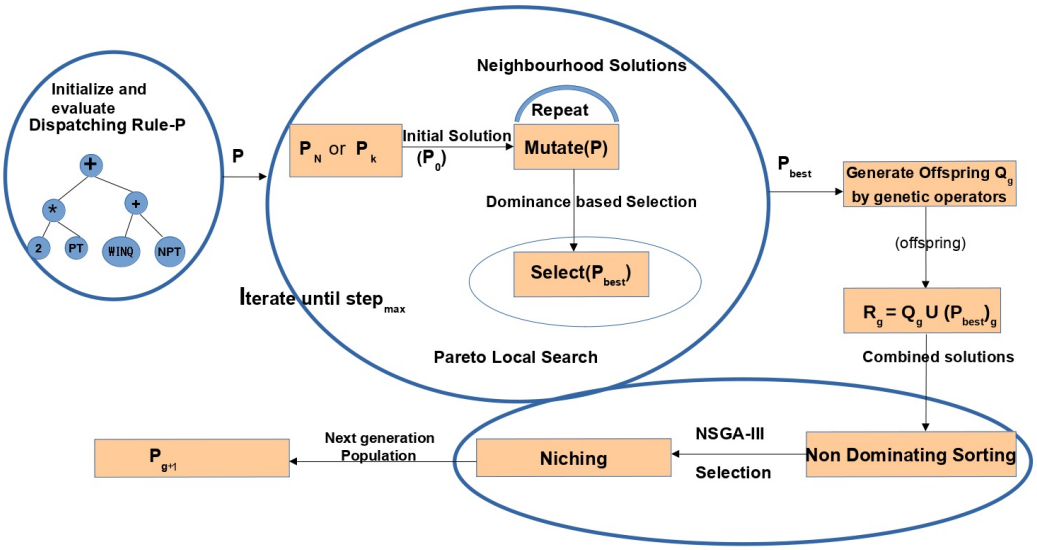


Figure 6.1: General Framework of GP-PLS-I.

Subsection 2.1.12 of Chapter 2.

In GP-PLS-I, first, K individuals are selected randomly from the population to form the archive. Then, each p in the archive (P_k) is selected for the neighborhood exploration by using the restricted mutation operator.

When the restricted mutation is applied to rule p, the algorithm randomly selects a node in p whose corresponding sub-tree has a depth of 2. This restriction tries to avoid significant change that makes the neighbor rules too different from the parent rules. A randomly generated depth-2 sub-tree then replaces the selected node and its sub-tree. With the help of this restricted mutation, GP-PLS-I can try to avoid significant changes that make the neighboring rules significantly different from the original rule [149]. This restricted mutation procedure is also described in Figure 6.2 and Algorithm 11. Exploring neighboring rules requires the exploration strategy, which determines the neighborhood's size for exploration and the selection strategy for best dispatching rules.

One can either explore the neighborhood entirely (best-improvement) [50]. Alternatively, only partially until the termination criterion is met [50].

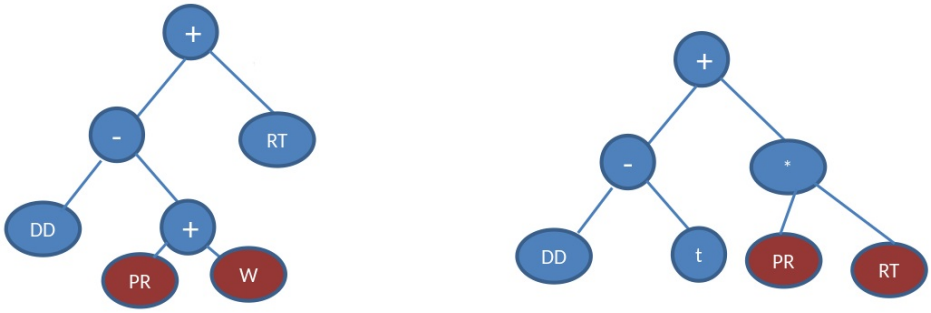


Figure 6.2: Examples of possible neighbor rules(shaded sub trees represented newly generated sub tree).

Algorithm 11: Restricted Mutation

Input : p

Output: p_{new}

1 Select a node at maximum depth of two in a p;
2 Randomly generate new subtree ;
3 $p_{new} \leftarrow$ replace a selected node with new subtree;
4 **return** (p_{new});

GP-PLS-I randomly samples a neighbor from the neighborhood repetitively until the maximum number of steps ($step_{max}$) is reached. Whenever the new program is better than the current program, it will replace the current program.

During the neighborhood exploration of rule p', p_{new} sampled from the neighborhood will be compared with p' (e.g., line 11 of Algorithm 10). Accordingly, the following two strategies are considered: (1) the scalarization strategy [79] and (2) the replacement strategy [26]. In the scalarization strategy, the objective vector of each rule is aggregated into a scalar using

weighted sum, i.e.

$$fit(x) = w_1 \cdot f_1(x) + w_2 \cdot f_2(x) + \cdots + w_m \cdot f_m(x), \tag{6.1}$$

where $w = (w_1, \ldots, w_m)$ is a random weight vector such that $w_i \geq 0$ ($\forall i = 1, \ldots, m$) and $w_1 + \cdots + w_m = 1$.

The replacement strategy is based on the dominance relation. When we compare two rules p_{new} and p'; there are three possible outcomes based on the replacement strategy:

1. If p_{new} dominates p', p' is replaced by p_{new}.
2. If p_{new} and p' are incomparable to each other, randomly choose one of them.
3. If p_{new} is dominated by p', do nothing.

By replacing rule p with rule p_{new} that dominates it in the archive, we can impose selection pressure on the archive and push it towards the Pareto-front.

In order to maintain a well-diversified collection of rules in the archive, we combine the GP-PLS-I with the niching mechanism used by NSGA-III (the detailed information can be seen in Subsection 2.1.5 of Chapter 2).

6.2.3 GP-PLS-II overview

GP-PLS-II algorithm is an extension of GP-PLS-I. GP-PLS-II uses the reference points and partitions the whole objective space into several sub-regions. The idea of decomposing the objective space has also been employed extensively in recent literature [31, 43]. This decomposition of the objective space determines the appropriate search direction of each solution. Thus solutions that belong to the same search direction are resided in the same sub-region of the objective space and called representatives of this subgroup. With the help of Equation (6.2), the number of representative solutions can be identified for each reference point. So, K_r best

solutions are selected according to their fitness values from each subspace. The number of solutions to be selected for neighborhood exploration from each sub-region is:

$$Number\ of\ Representatives(K_r) = \left(\frac{number\ of\ solution\ from\ each\ subspace}{Total\ population}\right) \times K. \tag{6.2}$$

Representatives who have the highest fitness value are given priority to enter the archive of the initial solution. Then, each individual in the archive is selected for the neighborhood exploration. The selection of the representatives from each sub-region guides the search toward the Pareto-front while guaranteeing good population diversity in the objective space. GP-PLS-II has two variations, the GP-PLS-II-Uniform and GP-PLS-II-Adaptive. GP-PLS-II-Uniform uses a set of uniform reference points that are defined by Das and Dennis's systematic approach [39].

For GP-PLS-II-adaptive, we adopt the adaptive reference points approach. For generating the adaptive reference points, we select the GP-MARP-NSGA-III algorithm from Section 5.5 of Chapter 5. GP-MARP-NSGA-III outperformed other adaptive reference points approaches (A-NSGA-III, NSGA-III-DRA).Therefore, we select GP-MARP-NSGA-III algorithm in GP-PLS-II. The details of the GP-PLS-II components and its framework are described below.

In GP-PLS-I, K solutions are selected randomly from the whole population. However, in GP-PLS-II, individuals are selected as representatives from each sub-region according to their local fitness value. In GP-PLS-II, we used the following steps for the selection of K solutions.

1. Combine the parent (P_g) and offspring (Q_g) population and obtain the combined population (R_g).

2. Use a decomposition-based approach to split the objective space into a number of independent sub-regions according to a set of reference points. Solutions associated with similar reference points having an identical search direction.

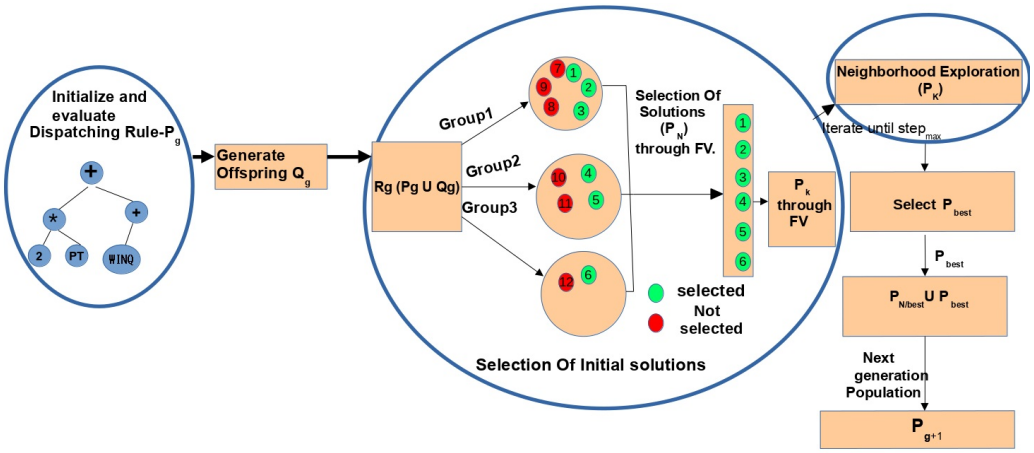


Figure 6.3: Framework of GP-PLS-II.

3. Assign fitness values to each solution and N solutions are selected based on their fitness values.

4. Choose K solutions from the N selected solutions as a representative according to their fitness value. Algorithm 12 outlines the selection process of solutions in GP-PLS-II and the complete workflow is shown in Figure 6.3.

For partitioning the objective spaces into many subspaces, GP-PLS-II generates reference points (either uniformly or adaptively). The generation of adaptive reference points is shown in Algorithm 8 of Chapter 5.

Through decomposition, the search direction of each solution can be determined. Two solutions, s_1 and s_2, have identical search directions if they are associated with the same reference point. Reference points are positive and inside the first quadrant, therefore, population R_g are normalized (see line 9 of Algorithm 12) before partitioned into $2N$ subpopulations R_{g1}, R_{g2},... R_{gN} by associating each individual with its closest reference point. The association of r is described in Figure 6.4.

Framework of GP-PLS-II

Algorithm 12: Solution selection in GP-PLS-II.

Input : A set of non-dominated solutions (rules) $\boldsymbol{P_g}$
Output: archive of selected solutions for neighbourhood

Apply genetic operators to $\boldsymbol{P_g}$ to generate offspring $\boldsymbol{Q_g}$;
foreach $Q \in \boldsymbol{Q}_g$ **do** Evaluate rule Q;
Combine $\boldsymbol{P_g}$ and $\boldsymbol{Q_g}$ ($\boldsymbol{R_g}$=$\boldsymbol{P_g} \cup \boldsymbol{Q_g}$);
Generate reference points $\boldsymbol{W}$ **for** *j =1 to* $\| \boldsymbol{R} \|$ **do**
    Calculate the ideal point Z_j^{min}=$min_{r \in R}\ f_j(r)$;
    Calculate the worst point Z_j^{max}=$max_{r \in R}\ f_j(r)$
end
for *i = 1 to PopSize* **do**
    $\hat{f(i)}$=$\frac{f_i - Z_j^{min}}{Z_j^{max} - Z_j^{min}}$;
end
foreach $r \in R_g$ **do**
    foreach $w \in W$ **do**
        compute the acute angle $\langle \hat{f(r)}, w \rangle$;
    end
    Assign $\hat{w} = w : argmin_{w \in W}\ \langle \hat{f(r)}, w \rangle$;
    /* Population Partition */;
    Assign $\theta_r = \langle \hat{f(r)}, w \rangle$;
    save r in $E(\hat{w})$
end
/* fitness of individual from each sub-region */;
foreach $w \in W$ **do**
    foreach $r \in E(\hat{w})$ **do**
        Compute the convergence criteria $C(r)$;
        Compute the diversity criteria $D(r)$;
        Compute the fitness of each individual $FV(r)$ by using equation (6.5)
    end
end
foreach $w \in W$ **do**
    Select solution according to the $FV(r)$ add selected solution from each subspace into the $archive$
end
return $archive$;

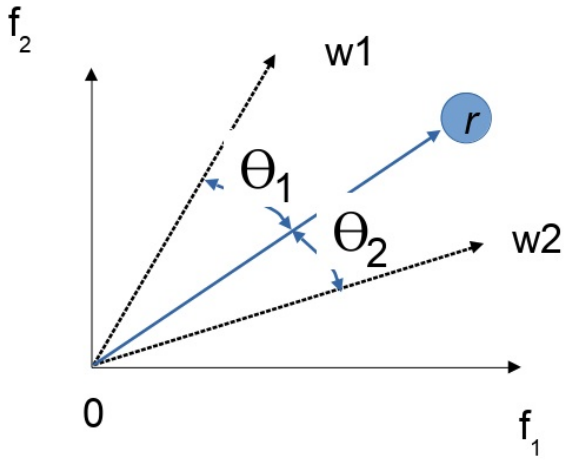


Figure 6.4: Example showing how to associate an individual r with a reference points. In this example, w_1 and w_2 are two unit reference points, $theta_1$ and $theta_2$ are the angles between r and w_1 and w_2, respectively. Since $theta_2 < theta_1$, the individual denoted by r is associated with reference points w_2.

The acute angle measures the association between individual and reference points. During the recent few years, the vector angle has attracted a high level of interest in evolutionary many-objective optimization [92, 30]. In our algorithm, the vector angle reflects the similarity of search directions between two individuals and latter, the angle information between two individuals in the objective space is used to maintain the diversity. The acute angle can be calculated as:

$$cos\theta_{i,j} = \frac{r_{i,j} \cdot w_{i,j}}{|r_{i,j}|}. \tag{6.3}$$

where $r_{i,j}$ is an individual from the combined population of the size 2N and $w_{i,j}$ is a reference point.

If an individual $r_{i,j}$ and $w_{i,j}$ have a minimal acute angle among all the reference points, $r_{i,j}$ becomes the member of the subpopulation $R_{g,k}$.

Once the population R_g is partitioned into 2N subpopulations, N solutions are selected through their fitness value. The selection criteria based on the fitness value (FV) are designed based on two sub-criteria: (1) the convergence criteria (d_1 in Figure 6.5) and (2) the diversity criteria (d_2 in Figure 6.5). d_1 is represented by the distance from solution $(r_{i,j})$ to the ideal point (Z^*) i.e, $\parallel r_{i,j} - Z^* \parallel$. Similarly, d_2 is represented by the inverse of the acute angle between $(r_{i,j})$ and $w_{i,j}$, i.e., $\theta_{i,j}$. In order to balance between the convergence criterion and the diversity criterion total FV of each individual can be formulated as a scalarization function:

$$FV = d1 + \frac{d2}{\theta_m}. \tag{6.4}$$

since our motivation is to find the solution on each reference point that is closest to the ideal point Z^*. Therefore, θ_m is used in equation (6.4) to

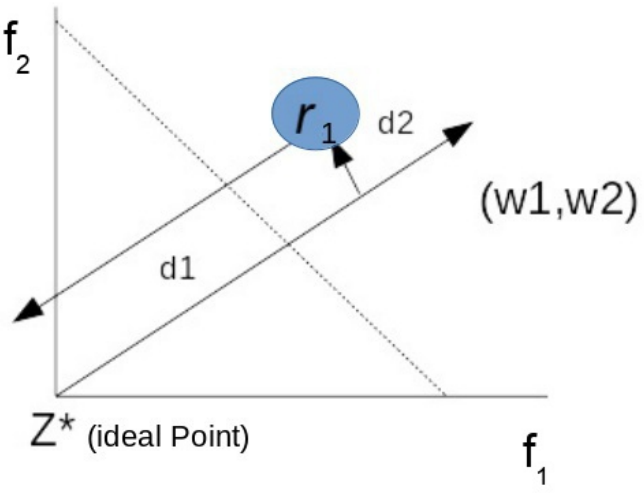


Figure 6.5: Distance measure in the context of minimization with respect to a reference direction.

normalize $\theta_{i,j}$. This angle normalization process is adopted from RVEA [31]. This process is meaningful when some of these reference points are sparsely distributed or densely distributed. As a result, angles between

the candidate solutions and the reference points are either extremely small or extremely large.

It is the best idea to apply high selection pressure on convergence during the exploration phase and push the population toward the Pareto-frontof the search process. However, during the exploitation phase, the constant pressure applies to diversity. Therefore, we introduce the penalty parameter $\frac{g}{g_{max}}$ in equation (6.5) which can better regulate the proportion of convergence and diversity information. So the new FV is expressed as:

$$FV = d1 + \frac{g}{g_{max}} * \frac{d2}{\theta_m}. \tag{6.5}$$

In the early stage, FV determines the convergence value (d1) because $g << g_{max}$, therefore d2 ≈ 0. However, when g approaches g_{max}, the penalty parameter gradually increases to emphasize the importance of the diversity criterion $\theta_{i,j}$. After getting the fitness values of each individual, then N solutions are selected. From N, K solutions are selected for the neighborhood exploration. Here parameter K is set identically as GP-PLS-I.

- **Neighborhood exploration:** Neighborhood explorations strategies govern the size of the explored neighborhood and the selection of neighboring solutions. GP-PLS-II supports a partial exploration of the neighborhood until the maximum number of steps $(step_{max})$ is reached. This number of steps is selected after the sensitivity analysis, which will discussed in subsection 6.3.1. The neighborhood solution is obtained from any given rule p using the restricted mutation operator during neighborhood exploration. The detailed information of the restricted sub-tree mutation can be found in Section 6.2.2.

- **Comparison:** The replacement strategy as discussed in Subsection 6.2.2 using the dominance relation to compare any two rules (p and p_{new}). The replacement strategy is also used in GP-PLS-II. It is obvious that replacing the current rule p with any neighborhood rule that

dominates p_{new} would help in the coverage of the Pareto-optimal solutions. Selection pressure is applied to the solutions by performing PLS schema on the population P_k and new population P_{best} is created. The P_{best} combined with $P_{N/best}$ and created a new population P_{g+1}.

Computational Complexity of the GP-PLS-II

To analyze the computational complexity of the GP-PLS-II, we consider the main steps in one generation in the main loop of Algorithm 12. Apart from genetic operations such as crossover and mutation, the main computational cost has resulted from the objective normalization, population partition (line 16 of Algorithm 12), calculation of fitness value (line 25 of Algorithm 12), and elitism selection (line 29 of Algorithm 12). The time complexity for the objective normalization (line 9 of Algorithm 12) is $\mathcal{O}(MN)$, where M is the objective number and N is the population size. The time complexity for the population partition of the 2N subpopulation is $\mathcal{O}(MN^2)$. In addition, the calculation of fitness value by using equation (6.4) holds a computational complexity of $\mathcal{O}(MN^2)$. Further, for elitism selection, computational resources are mainly consumed by both convergence and diversity selection. The computational complexity of elitism selection is $\mathcal{O}(MN^2)$ and $\mathcal{O}(N^2)$ in the worst-case scenario.

To summarize, apart from the genetic variations, the worst-case, that is, all the 2N individuals get trapped into one subspace and other subspaces do not contain any member, the overall computational complexity of GP-PLS-II within one generation is $\mathcal{O}(MN^2)$, which is the same as the average complexity.

6.3 Design of experiment

The JSS benchmark which is a Taillard (TA) static JSS benchmark instances (the detailed information can be seen in Section 3.1 of Chapter 3) is se-

lected as a testbed. In the experiments, we considered four potentially conflicting objectives: (1) the mean flowtime (Obj1) (see equation (2.1) of Chapter 2), (2) maximal flowtime (Obj2) (see equation (2.2) of Chapter 2), (3) mean weighted tardiness (Obj3) (see equation (2.6) of Chapter 2) , and (4) maximal weighted tardiness (Obj4) (see equation (2.7) of Chapter 2) which have been defined in Section 3.2 of Chapter 2.

The terminal set and function set for the tree-based GP have been described in Section 3.2 of Chapter 3. The crossover, mutation, and reproduction rates are set identical to Subsection 2.1.2 of Chapter 4.

In the experiments, the two commonly used measures in multi-objective optimisation, i.e. IGD [206] and HV [212] are used to compare the algorithms (see section of in Section 3.3 of Chapter 3).

6.3.1 Sensitivity analysis

In a hybridized algorithm, it is important to understand how to divide the available computation time between the local search and the global search. In order to prevent the local search from spending almost all available computation time, we decide to use the partial strategy, which restricts the number of iteration in the local search. If we use a very small value of $step_{max}$ (e.g., $step_{max}$=1), the local search procedure may be terminated sooner than desired. On the contrary, if we use a large value of $step_{max}$ (e.g., $step_{max}$=10), the local search procedure tends to evaluate more solutions than necessary.

We need to carefully adjust the computation time spent by the local search procedure in our hybrid algorithm because of the above. Therefore, in this experiment, we examined different combinations of the parameters where population size is equal to 1000. These combinations are: $(K, steps_{max}, generations)$ = (1000,3,25), (500,2,50), and (250,4,50). The Sensitivity analysis applies to GP-PLS-I and selecting parameters are later used in the GP-PLS-I algorithm. We also used selected parameters in GP-

Table 6.1: The mean and standard deviation over the average HV and IGD values on training instances of the compared algorithms in the four-objective experiment.

HV ($\bar{x} \pm \sigma$)					
Comb1-(1000,3,25)		Comb2-(500,2,50)		Comb3-(250,4,50)	
GP-PLS-I-s	GP-PLS-I-r	GP-PLS-I-s	GP-PLS-I-r	GP-PLS-I-s	GP-PLS-I-r
0.634(0.013)	0.684(0.015)	0.630(0.020)	0.676(0.015)	0.690(0.018)	0.705(0.016)
IGD ($\bar{x} \pm \sigma$)					
GP-PLS-I-s	GP-PLS-I-r	GP-PLS-I-s	GP-PLS-I-r	GP-PLS-I-s	GP-PLS-I-r
0.00131(0.00017)	0.00127(0.00012)	0.00134(0.00017)	0.00130(0.00024)	0.00127(0.00013)	0.00122(0.00012)

PLS-II because GP-PLS-II is an extension of GP-PLS-I. The three-parameter settings have the same total number of fitness evaluations (100000). For a fair comparison, the number of fitness evaluations is kept identical to GP-NSGA-III. In the sensitivity analysis, 30 independent runs were performed to produce 30 final sets of dispatching rules for each combination of the parameters.

For the case of (1000,3,25), GP-PLS-I uses the whole population during the local search with three $step_{max}$ for exploring the neighborhood solutions. For the case of (500,2,50), GP-PLS-I can explore the solution space very well through 50 generations of evolution but has a small number of local searches. In contrast with the first two parameter combinations, (250,4,50) has a proper balance between global search (50 generations) and local search (4 steps during the local search) capabilities.

From the results summarized in Table 6.1, we found that the total number of generations and the maximum number of local search steps highly influenced the performance of GP-PLS-I and their computational time. They together provide varied trade-offs between global and local searches in GP-PLS-I.

The result showed that GP-PLS-I could not search the solution space extensively with a small number of generations in (1000,3,25). On the other hand, if GP-PLS-I cannot perform a sufficient number of local search steps in (500,2,50), the power of local search cannot be effectively utilized.

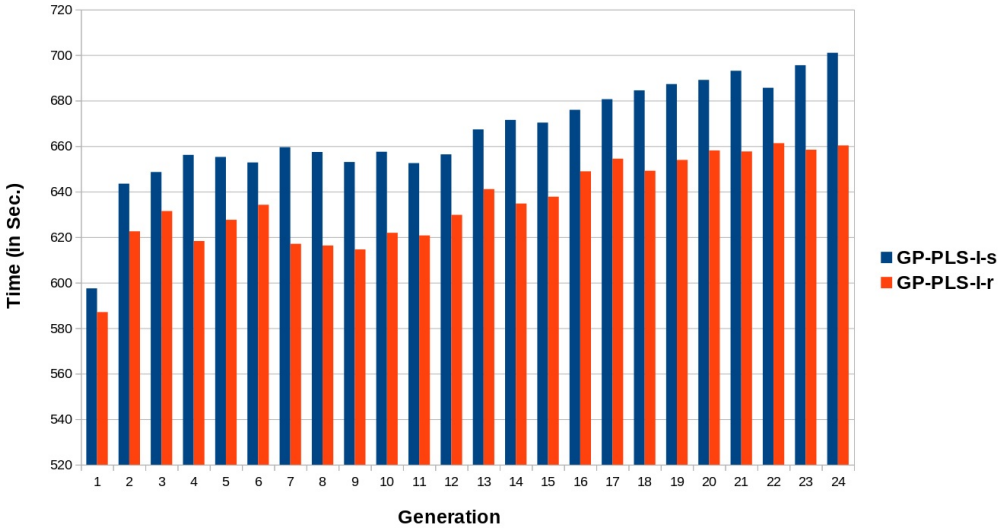


Figure 6.6: Computational time of whole population

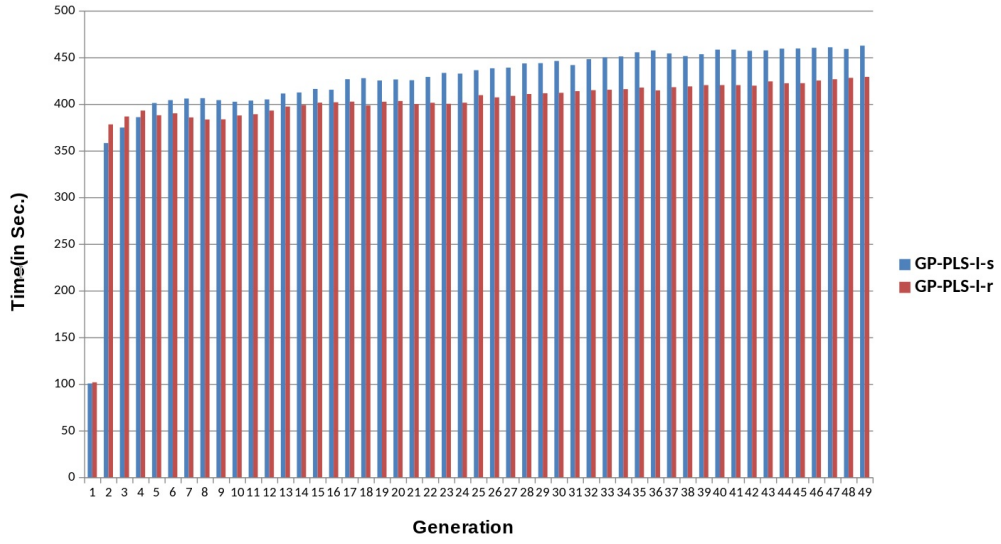


Figure 6.7: Computational time of sub-population.

Table 6.1 shows that a combination (250,4,50) significantly performed bet-

ter in terms of HV and IGD as compared to the other two combinations for PLS.

Figures 6.6 and 6.7 reveal that the local search on the whole population is more computationally expensive than the subset of solutions. For GP-PLS-I to achieve excellent performance, we select (250,4,50) in the subsequent experiments. GP-PLS-II is an extension of GP-PLS-I. Therefore, GP-PLS-II will also select similar parameters of GP-PLS-I.

6.4 Results and discussions

6.4.1 Results

Table 6.2: The mean and standard deviation over the average HV and IGD values on training instances of the compared algorithms in the four-objective experiment.

HV ($\bar{x} \pm \sigma$)				
GP-NSGA-III	GP-PLS-I-s	GP-PLS-I-r	GP-PLS-II-U	GP-PLS-II-A
0.68850(0.0221)	0.69048(0.0160)	0.70513(0.0130)	0.70967(0.0115)	**0.7133(0.0102)**
IGD ($\bar{x} \pm \sigma$)				
GP-NSGA-III	GP-PLS-I-s	GP-PLS-I-r	GP-PLS-II-U	GP-PLS-II-A
0.00125(0.00015)	0.00127(0.00013)	0.00122(0.00012)	0.00126(0.00016)	**0.00123(0.000059)**

Tables 6.2 shows the mean and standard deviation of the training performance in terms of HV and IGD of the rules obtained by GP-NSGA-III, GP-PLS-I-s, GP-PLS-I-r, GP-PLS-II-Uniform (GP-PLS-II-U), and GP-PLS-II-Adaptive (GP-PLS-II-A). Here GP-PLS-I-s refers to the variation of GP-PLS-I where the scalarization approach is used for selection in Algorithm 10. On the other hand, GP-PLS-I-r represents the variation where the replacement strategy is used for selection in Algorithm 10.

For each algorithm in the experiment, 30 GP runs are conducted to obtain 30 sets of dispatching rules. Then, the rules are tested on the 40 test instances. The Wilcoxon rank-sum test [195], with the significance level of

0.05 is applied to the HV and IGD of the Pareto-front evolved by the five compared algorithms.

Table 6.2 reveals that GP-PLS-II-A performs significantly better than other competing algorithms in terms of both HV and IGD. Table 6.2 shows that GP-PLS-I-r is also highly competitive with GP-PLS-II-U. Therefore, we can confirm that the replacement strategy is more effective than the scalarization strategy. Table 6.2 further shows that the selection of an initial solution based on FV in PLS improves the algorithm's performance compared to those that adopt the random selection. Therefore, both versions of GP-PLS-II performed significantly better than GP-PLS-I in terms of HV and IGD.

Tables 6.3 and 6.4 show the mean and standard deviations of the test performance on each of the 40 test instances. In the case of HV, GP-PLS-II-A performed the best in 17 instances out of the 40 test instances. GP-PLS-II-U outperformed the other algorithms in 8 instances, GP-PLS-I-r performed the best in 5 instances, and GP-PLS-I-s outperformed other algorithms in 2 instances. GP-NSGA-III performed the best only on one instance.

Regarding IGD, GP-PLS-II-A performed significantly better than other algorithms in 14 instances out of the 40 test instances. GP-PLS-II-U outperformed the other algorithms in 9 instances. GP-PLS-I-r performed the best in 4 test instances. In contrast, GP-NSGA-III performed the best in 4 instances. GP-PLS-I-s outperformed the other algorithms in only 2 instances.

Tables 6.3 and 6.4 show that GP-PLS-I-r performed significantly better in more instances than GP-PLS-I-s and GP-NSGA-III. The results indicate that GP-PLS-I-r is more effective in utilizing the dominance relation during the local search. Upon taking a closer look at Tables 6.3 and 6.4, it can be observed that GP-PLS-II algorithms not only performed well on small-scale problem instances but also on larger and more challenging instances in terms of HV and IGD. Tables 6.3 and 6.4 further show that GP-PLS-II al-

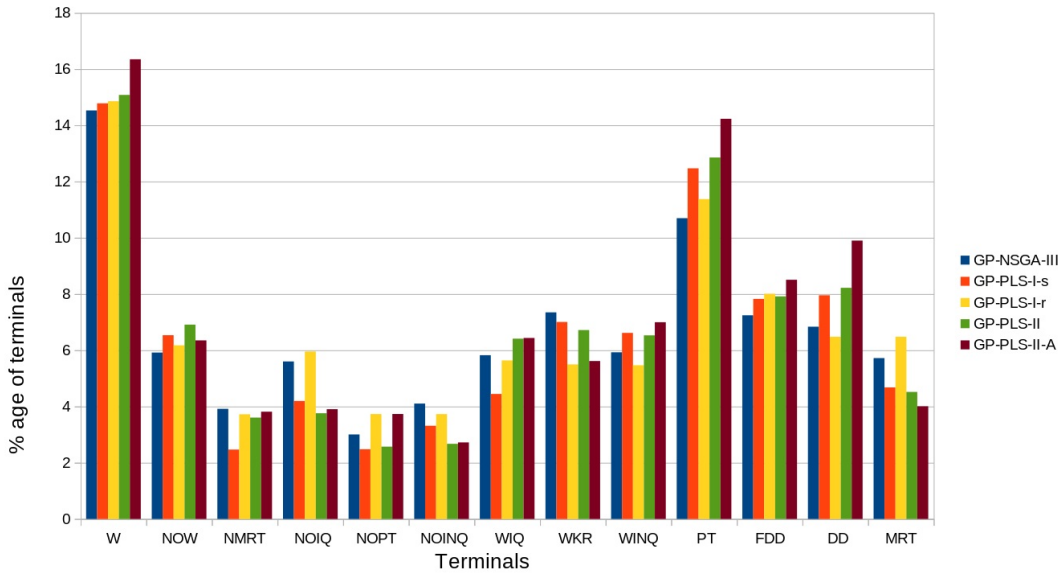


Figure 6.8: Frequency of terminals in GP-NSGA-III, GP-PLS-I-s, GP-PLS-I-r, GP-PLS-II, and GP-PLS-II-A.

gorithms performed significantly better than GP-PLS-I-r and GP-PLS-I-s. The results indicate the effectiveness of selecting solutions based on the FV for neighborhood exploration. The results also demonstrate that overall, GP-PLS algorithms performed significantly better than GP-NSGA-III.

6.4.2 Discussion

Analysis of dispatching rules

Previous results have shown that GP-PLS-II is a very effective approach to discovering dispatching rules for JSS. This section will explore its behaviors to understand how it can effectively search for dispatching rules. The bar chart in Figure 6.8 shows the percentage of terminals in evolved rules from each algorithm. Useful terminals for optimizing flowtime and tardiness objectives have been discussed in Subsection 4.5.3 of Chapter 4. It can be seen in Figure 6.8 that more than 10 percent of evolved rules from each

Table 6.3: The mean and standard deviation over the HV values on the test instances of the compared algorithms.

ID	#J_#M	GP-NSGAIII	GP-PLS-I-s	GP-PLS-I-r	GP-PLS-II-U	GP-PLS-II-A
1	15_15	.1868(.0368)	.1698(.0125)	**.2723(.0145)**	.2631(.0173)	.1060(.0632)
2	15_15	.3296(.0182)	.3070(.0145)	.4091(.0110)	.4141(.0161)	.3271(.0782)
3	15_15	.2175(.0225)	.2683(.0125)	.2508(.0127)	.3191(.0371)	**.3955(.0671)**
4	15_15	.3159(.0183)	.1259(.0133)	**.4561(.0091**	.1771(.0087)	.396(.0107)
5	15_15	.2370(.0202)	.1799(.0302)	.3479(.0192)	.3360(.0462	.2719(.0577)
6	20_15	.4147(.0176)	.4024(.0186)	.4280(.0139)	.3166(.0162)	**.4719(.0537)**
7	20_15	.2396(.0683)	.2551(.0695)	.3146(.0683)	.3636(.0613)	**.3779(.0582)**
8	20_15	.2935(.0201)	**.3201(.0253)**	.2158(.0501)	.2508(.0590)	.2969(.0612)
9	20_15	.1787(.0198)	.3232(.0198)	.1606(.0219)	**.4021(.0832)**	.2019(.0502)
10	20_15	.3180(.0151)	.3847(.0141)	.2743(.0366)	.3533(.0167)	**.4187(.0609)**
11	20_20	.2114(.0038)	.2214(.0138)	**.3087(.0118)**	.0086(.0064)	.2218(.0605)
12	20_20	.3652(.0133)	.2452(.0123)	.3206(.0134)	.3206(.0134)	**.3974(.0159)**
13	20_20	.4540(.0151)	.4541(.0326)	.4550(.0221)	**.4971(.0163)**	.2469(.0175)
14	20_20	.1658(.01103)	.2945(.01403)	.3318(.0118)	**.3658(.0558)**	.2146(.0155)
15	20_20	.1742(.0225)	.1840(.0199)	.1352(.0125)	.1558(.0158)	**.3704(.0119)**
16	30_15	.2964(.0300)	.3508(.0234)	.3198(.0110)	**.3733(.0428)**	.2008(.0157)
17	30_15	.3741(.0096)	.3385(.0229)	.4076(.0093)	.3915(.0529)	**.4729(.0145)**
18	30_15	.3825(.0233)	.3710(.3032)	.3274(.0103)	**.4571(.0279)**	.3988(.0139)
19	30_15	**.4353(.0126)**	.3808(.0197)	.3988(.0146)	.3495(.0264)	.4119(.0302)
20	30_15	.3800(.0312)	.3801(.0312)	.3762(.0212)	.3331(.0296)	.3198(.0339)
21	30_20	.1983(.0657)	.3340(.0792)	.2190(.0492)	.3635(.0719)	**.3768(.0449)**
22	30_20	.2385(.0472)	.2954(.0470)	.3177(.0372)	.3095(.0417)	**.3599(.0602)**
23	30_20	.2020(.0398)	.2672(.0410)	.3675(.0294)	.3510(.0267)	.3468(.0449)
24	30_20	.4420(.0503)	.4652(.0443)	.4529(.0174)	.3510(.0267)	.3468(.0449)
25	30_20	.3372(.0477)	.3854(.0396)	.2864(.0427)	.4555(.0359)	**.4947(.0418)**
26	50_15	.4563(.0417)	.4672(.0170)	.4872(.0270)	.5905(.0275)	**.6439(.0338)**
27	50_15	.5610(.0361)	.5685(.0304)	.5555(.0304)	.5510(.0529)	**.5849(.0238)**
28	50_15	.4598(.0398)	.4966(.0250)	.4798(.0333)	.5250(.0619)	**.6029(.0182)**
29	50_15	.5749(.0372)	.5702(.0251)	.4323(.0413)	.4418(.0274)	**.6260(.0241)**
30	50_15	.4510(.0406)	.4310(.0333)	.4732(.0240)	**.5435(.0269)**	.5049(.0253)
31	50_20	.5190(.0424)	.4477(.0295)	.4870(.0433)	.5124(.0429)	**.5830(.0171)**
32	50_20	.4354(.0476)	**.5705(.0476)**	.4996(.0375)	.4285(.0429)	.5166(.0211)
33	50_20	.4427(.0266)	.4426(.0838)	.5165(.0366)	.4525(.0279)	.5206(.0234)
34	50_20	.4108(.0384)	.3945(.0262)	**.4911(.0332)**	.6015(.0239)	.4579(.0253)
35	50_20	.4421(.0349)	.4138(.0222)	.4140(.0165)	**.6610(.0189)**	.5049(.0253)
36	100_20	.6054(.0179)	.59169(.0093)	**.6228(.0222)**	.5685(.0229)	.4571(.0343)
37	100_20	.6584(.0142)	.6678(.0101)	.5857(.0152)	.5735(.0182)	**.6989(.0188)**
38	100_20	.6152(.0196)	.6175(.0136)	.5525(.0111)	**.6791(.0267)**	.6015(.0173)
39	100_20	.6495(.0185)	.6515(.0191)	.6695(.0103)	.6305(.0229)	**.6885(.0188)**
40	100_20	.6830(.0158)	.6322(.0100)	.6467(.0104)	.6913(.0339)	.6165(.0199)

Table 6.4: The mean and standard deviation over the IGD values on the test instances of the compared algorithms.

ID	#J_#M	GP-NSGA-III	GP-PLS-I-s	GP-PLS-I-r	GP-PLS-II-U	GP-PLS-II-A
1	15_15	.01897(.0008)	.02748(.00011)	.01479(.00012)	**.0119(.00017)**	.0299(.00043)
2	15_15	.0127(.0005)	.0124(.0004)	.0117(.0003)	**.0098(.00015)**	.0133(.00023)
3	15_15	.0173(.0008)	.0126(.0006)	.0137(.0002)	.0115(.0005)	**.0112(.0026)**
4	15_15	.0162(.0007)	.0260(.0006)	**.0135(.0002)**	.0300(.0006)	.0206(.0005)
5	15_15	.0193(.0005)	.0200(.0003)	**.0085(.0007)**	.0119(.0001)	.0115(.0005)
6	20_15	**.0079(.0004)**	.0147(.0001)	.0124(.0007)	.0111(.0001)	.0300(.0006)
7	20_15	.0250(.0013)	.0109(.0012	.0121(.0014)	.0010(.0009)	**.0093(.0002)**
8	20_15	.0133(.0001)	.0156(.0012)	.0147(.0013)	.0116(.0016)	**.0106(.0002)**
9	20_15	.0201(.0001)	.0108(.0005)	.0142(.0003)	**.0078(.0008)**	.0085(.0001)
10	20_15	.0129(.0002)	**.0077(.0001)**	.0128(.0003)	.0149(.0002)	.0108(.0001)
11	20_20	.0094(.0006)	.0090(.0001)	.0091(.0003)	.0221(.0007)	**.0119(.0014)**
12	20_20	.0217(.0004)	.0168(.0001)	.0147(.0006)	.0109(.0002)	**.0008(.0018)**
13	20_20	.0155(.0001)	.0112(.0005)	.0100(.0004)	.0009(.0001)	.0008(.0002)
14	20_20	.0246(.0009)	.0174(.0005)	.0216(.0004)	**.0165(.0008)**	.0283(.0029)
15	20_20	.0191(.0008)	.0315(.0003)	.0289(.0004)	.0083(.0001)	**.0024(.0008)**
16	30_15	.0111(.0006)	.0090(.0002)	.0070(.0008)	**.0013(.0001)**	.0018(.0006)
17	30_15	.0061(.0003)	**.0047(.0008)**	.0077(.0007)	.0057(.0001)	.0058(.0003)
18	30_15	.0060(.0006)	.0092(.0008)	.0063(.0007)	.0068(.0002)	**.0038(.0007)**
19	30_15	.0065(.0004)	.0059(.0008)	**.0054(.0007)**	.0064(.0002)	.0063(.0007)
20	30_15	**.0019(.0005)**	.0054(.0003)	.0065(.0002)	.0092(.0007)	.0062(.0007)
21	30_15	.0094(.0022)	.0097(.0013)	.0125(.00012)	.0122(.0003)	**.0079(.0009)**
22	30_20	.0098(.0010)	.0103(.0006)	.0095(.00014)	.0079(.00017)	**.0061(.0005)**
23	30_20	.0077(.00004)	.0074(.00006)	**.0063(.00024)**	.0082(.0015)	.0127(.0019)
24	30_20	.0069(.00014)	.0056(.00016)	.0085(.00021)	**.0052(.0008)**	.0058(.0001)
25	30_20	.0056(.00012)	.0071(.00005)	.0074(.00014)	.0057(.0001)	**.0067(.0005)**
26	50_15	.0057(.00009)	.0058(.00005)	0054(.00044)	.0048(.00007)	.0043(.0013)
27	50_15	**.0038(.00029)**	0054(.00013)	.0042(.00041)	.0045(.0006)	.0048(.0007)
28	50_15	.0036(.00010)	.0035(.00033)	.0040(.00051)	.0041(.0005)	.0045(.0005)
29	50_15	**0035(.00032)**	.0062(.00034)	.0045(.00031)	.0066(.00033)	.0043(.0003)
30	50_15	.0043(.00010)	.0045(.00044)	.0042(.00011)	.0049(.0008)	.0053(.0008)
31	50_20	.0047(.00029)	.0046(.00003)	.0052(.00018)	**.0041(.0007)**	.0044(.0002)
32	50_20	.0040(.00009)	.0054(.00006)	.0041(.00008)	.0062(.0005)	.0051(.0005)
33	50_20	.0054(.00001)	.0056(.00002)	.0044(.00008)	.0027(.0002)	**.0018(.0008)**
34	50_20	.0037(.00005)	.0035(.00008)	.0033(.00009)	.0036(.0004)	.0049(.0006)
35	50_20	.0032(.00015)	.0034(.00007)	.0031(.00009)	**.0030(.0002)**	.0047(.0004)
36	100_20	.0027(.00020)	.0032(.00001)	.0021(.00067)	.0039(.0006)	.0023(.0004)
37	100_20	.0034(.00018)	.0031(.00001)	.0027(.00017)	.0035(.0004)	**.0025(.0002)**
38	100_20	.0044(.00028)	0037(.00004)	0040(.00008)	**.0028(.0004)**	.0032(.0003)
39	100_20	.0026(.00020)	.0028(.00001)	.0024(.00067)	.0028(.0003)	**.0018(.0001)**
40	100_20	0018(.00001)	.0019(.00002)	.0022(.00009)	.0023(.0003)	**.0013(.0002)**

algorithm have W and PT terminals. According to the literature [70], MRT, PT, WKR, WINQ, NOINQ, FDD, NOPT are useful terminals (relevant) for optimizing flowtime objectives. Specifically, PT, WINQ, and WKR are the most important three terminals for optimizing flowtime objective. On the other hand, WINQ, NOINQ, NOPT, W, PT, MRT, DD are the useful terminals for optimizing tardiness objectives. Of these, PT, DD, and W are the most useful terminals for optimizing tardiness objectives.

It can be seen from Figure 6.8 that the number of occurrences of W, WINQ, PT, DD, MRT, NOPT and FDD terminals are higher in the local search algorithms (GP-PLS-I-s, GP-PLS-I-r, GP-PLS-II, and GP-PLS-II-A) than in the GP-NSGA-III algorithms. This is because all the GP-PLS algorithms enhanced the exploitation ability and evolved significantly better rules as compared to the other algorithms in terms of HV and IGD. Therefore, these algorithms selected the rules which are well-optimized. As a result, there are more chances of occurrences of useful terminals in GP-PLS algorithms. Further, it can also be seen from Figure 6.8 that GP-PLS-II-A has more useful terminals (W, PT, DD, FDD, and WINQ) than GP-PLS-I-s, GP-PLS-I-r, GP-PLS-II. This analysis shows the effectiveness of adaptive reference points with PLS.

Figure. 6.9 shows the length of rules from each generation in GP-NSGA-III, GP-PLS-I-s, GP-PLS-I-r, GP-PLS-II. In most of the generations, GP-PLS-II produces shorter rules as compared to GP-PLS-I-r, GP-PLS-I-s, and GP-NSGA-III. In fact, all the hybridized algorithms with PLS have relatively short rules as compared to GP-NSGA-III. It is noted that the program lengths in Figure. 6.9 average length of the best rules of each independent run. This is another advantage of PLS as the short rules are easier to analyze and interpret. All PLS-II requires less time for evaluations as compared to GP-NSGA-III. This, in turn, may affect the computational time of the PLS-algorithm.

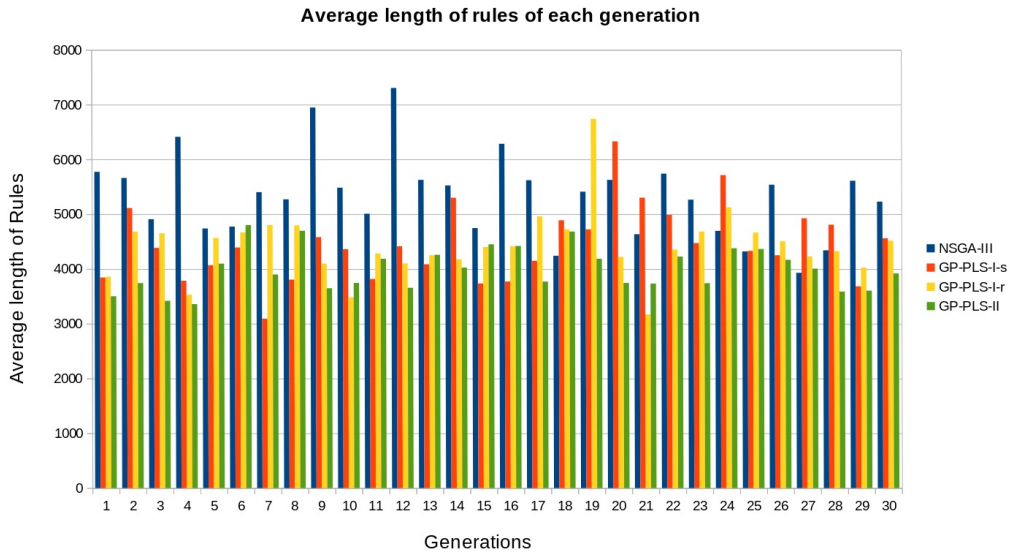


Figure 6.9: Length of rules from each generation in GP-NSGA-III, GP-PLS-I-s, GP-PLS-I-r, GP-PLS-II.

Further analysis

To further investigate how PLS affects the GP search process, we plotted (a) the average HV and IGD of non-dominated solutions evolved by GP-PLS across multiple generations in Figures. 6.10 and 6.11, (b) parallel coordinate plots of non-dominated solutions evolved by GP-PLS and GP-NSGA-III algorithms on one problem instance in Figures 6.12 (a) to 6.16 (b), and (c) box-plots of the HV and IGD values on different test instances in Figures 6.19 (a) to 6.22 (b).

Figures 6.10 and 6.11 reveal that GP-PLS-II-A has better convergence curves in terms of both HV and IGD than other compared algorithms. Figures 6.10 and 6.11 also show that both algorithms of GP-PLS-II achieved better performance than the GP-PLS-I and GP-NSGA-III in terms of HV and IGD. These results reveal that the selection of the solutions based on FV (convergence and diversity) can improve the GP-PLS algorithm's overall performance.

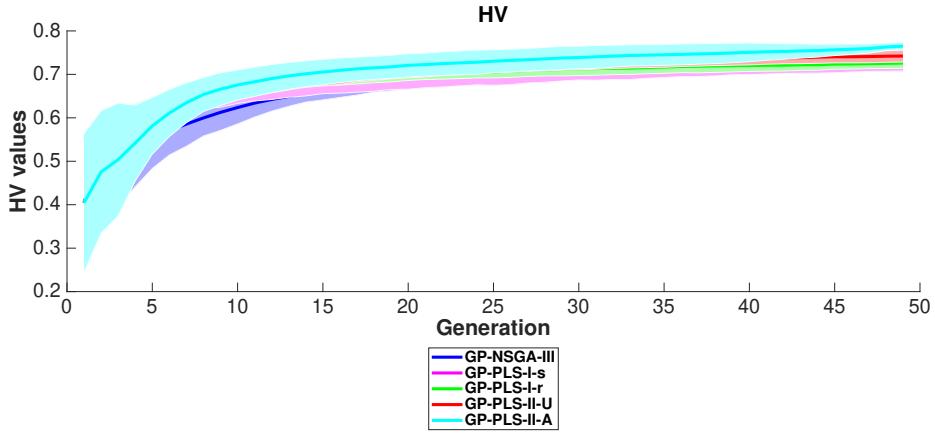


Figure 6.10: The curves of the average number of HV value of the non-dominated solutions on the training set during the 30 independent GP runs.

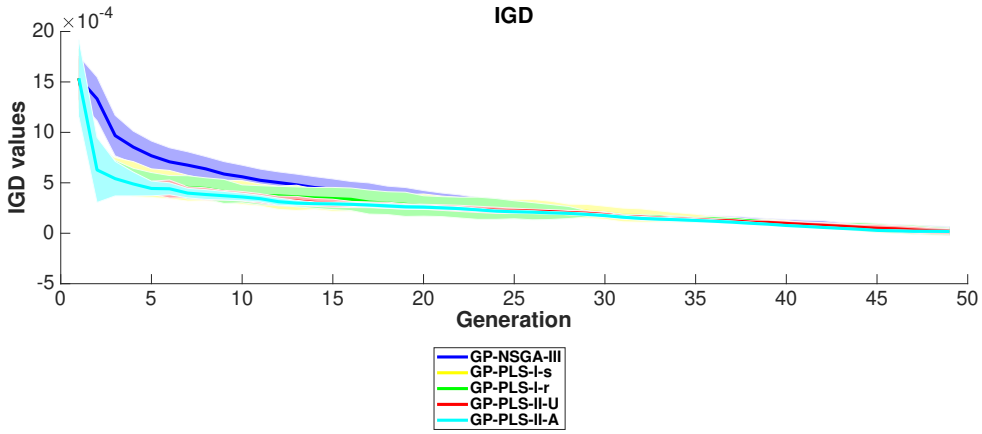
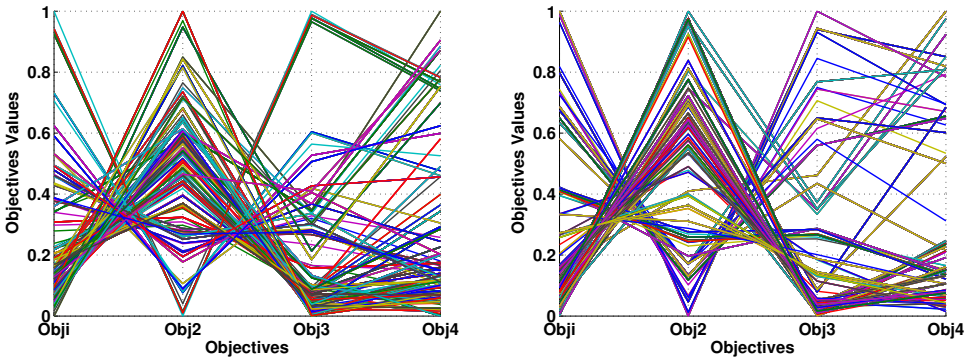


Figure 6.11: The curves of the average number of IGD value of the non-dominated solutions on the training set during the 30 independent GP runs.

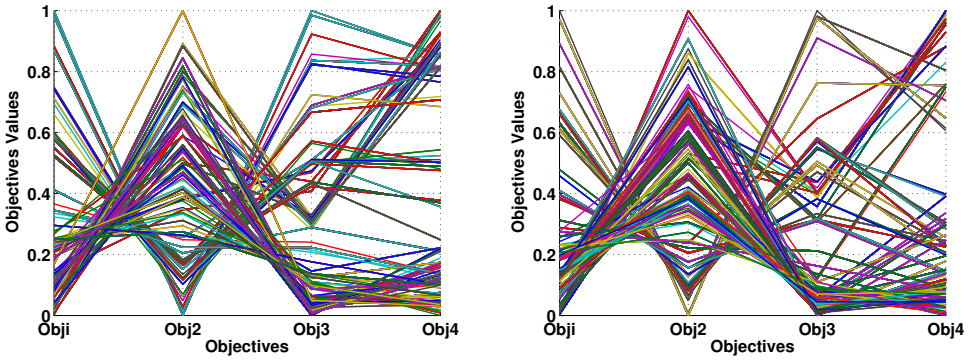
The parallel coordinate plots in Figures 6.12 (a) to 6.16 (b) depict the non-dominated set of dispatching rules obtained respectively by GP-NSGA-III. Figures 6.12 (a), 6.13 (a), 6.14 (a), 6.15 (a), and 6.16 (a) show that GP-PLS-I-r successfully evolved rules with better coverage for the objec-

tives (on mean flowtime and maximum weighted tardiness) in generation 10. On the other hand, it can be seen in Figures 6.12 (b), 6.13 (b), 6.14 (b), 6.15 (b), and 6.16(b) that rules evolved by GP-PLS-II-A are more diversified to cover a much wider range of the objective space in generation 50 than other compared algorithms. We can also observe that GP-PLS-II-U and GP-PLS-I-r are also well-diversified as compared to GP-NSGA-III and GP-PLS-I-s.



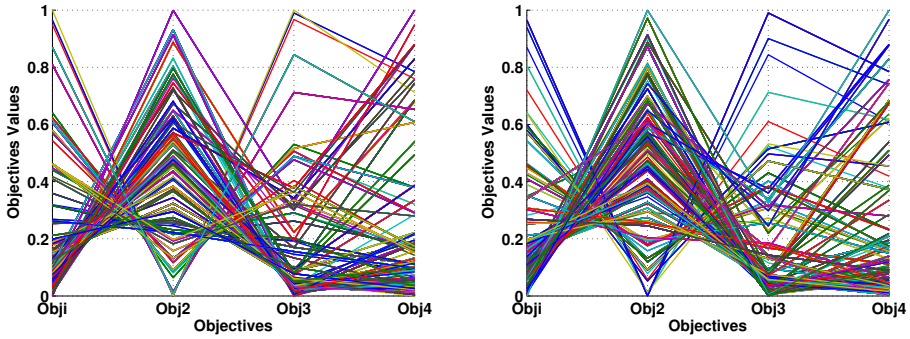
(a) Generation 10. (b) Generation 50.

Figure 6.12: Parallel coordinate plot of GP-NSGA-III.



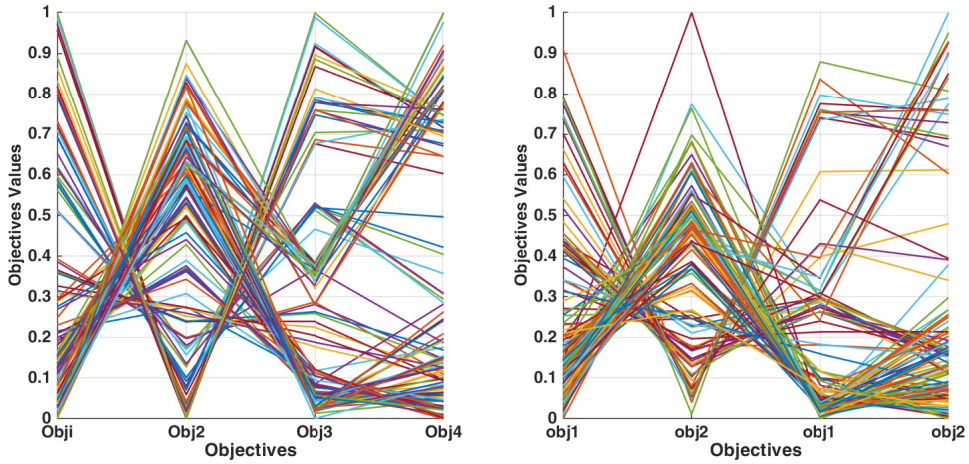
(a) Generation 10. (b) Generation 50.

Figure 6.13: Parallel coordinate plot of GP-PLS-I-s.



(a) Generation 10. (b) Generation 50.

Figure 6.14: Parallel coordinate plot of GP-PLS-I-r.

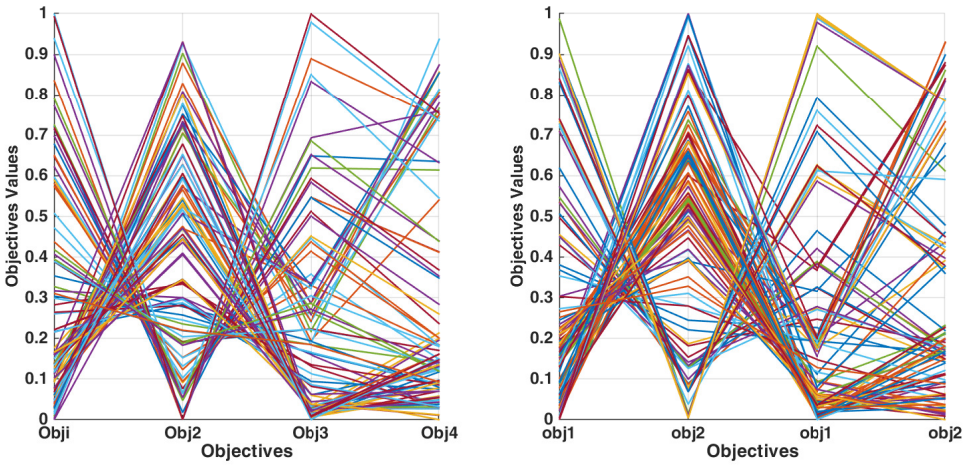


(a) Generation 10. (b) Generation 50.

Figure 6.15: Parallel coordinate plot of GP-PLS-II-U.

Figures 6.17 and 6.18 show the distribution of solutions of GP-PLS-II-A and GP-PLS-II-U to understand how GP-PLS-II-A performs better than GP-PLS-II-U. These figures show that GP-PLS-II-A has widely distributed optimal solutions than uniformly distributed decomposition-based algorithms.

Figures 6.19 (a) to 6.22 (b) show the box plots of the HV and IGD values



(a) Generation 10. (b) Generation 50.

Figure 6.16: Parallel coordinate plot of GP-PLS-II-A.

on different test instances. Figure 6.19 (a) shows a small instance where GP-PLS-II-A significantly outperforms GP-PLS-II-U in terms of HV. The box plot of GP-PLS-II-A is comparatively taller than GP-PLS-II-U which shows that the area occupied by non-dominated solutions is higher than GP-PLS-II-U. Further, the long upper whisker reveals that most of the HV values are close to the positive quartile region in GP-PLS-II-A. On the other hand, GP-PLS-II-U also has long upper whiskers with outliers.

Figure 6.19 (b) shows the box plot on one of the complex instances with a larger number of machines where GP-PLS-II-A is significantly better than GP-PLS-II-U in terms of HV. The box plot shows that GP-PLS-II-A produces higher HV values above the central region. Further, the box plot for GP-PLS-II-U is comparatively taller than GP-PLS-II-A. This result indicates that the area occupied by the non-dominated solutions is more significant than GP-PLS-II-A.

Figures 6.20 (a) and 6.20 (b) show the box plots of those instances where GP-PLS-II-U performs significantly better than GP-PLS-A in terms of HV. Figures 6.20 (a) and 6.20 (b) depict the box plot, showing that most of the

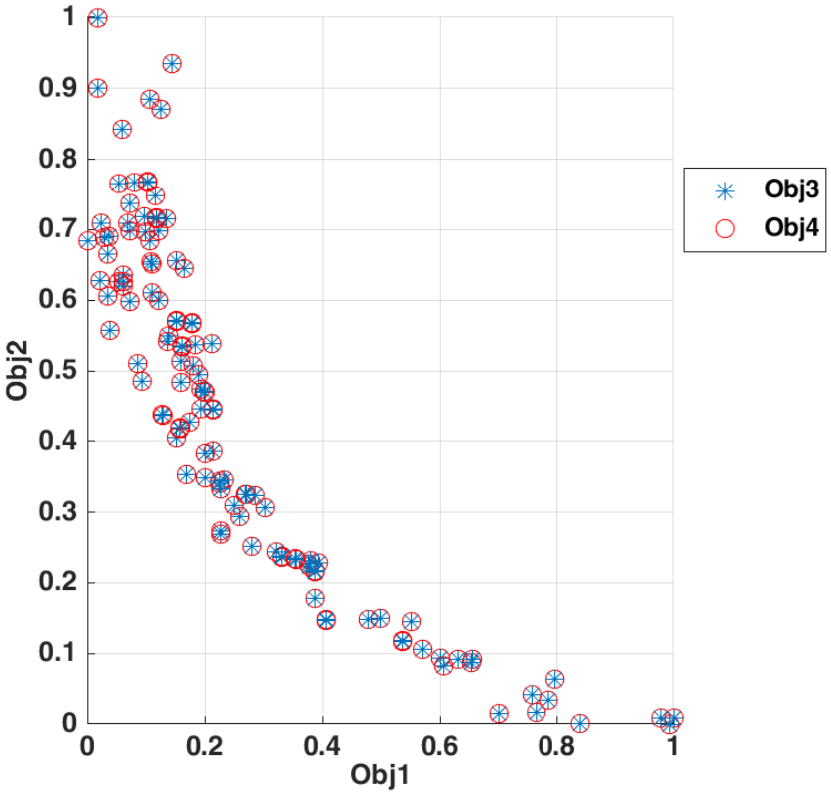


Figure 6.17: Distribution of solutions of GP-PLS-II-U on instance 26.

HV values are close to the positive quartile region for GP-PLS-II-U. On the other hand, GP-PLS-II-A also has a long upper whisker with outliers.

Figures 6.21 (a) to 6.21 (b) show that the box-plots of IGD values on smallest and complex test instances, respectively. Figure 6.21 (a) shows that GP-PLS-II-A significantly outperforms GP-PLS-II-U in terms of IGD. The box plot of GP-PLS-II-A is comparatively lower than GP-PLS-II-U. Figure 6.21 (b) also indicates that the Pareto-optimal solutions are much closer to the Pareto-front as compared to Pareto-optimal solutions obtained by GP-PLS-II-U.

Figures 6.22 (a) and 6.22 (b) show the box plots of those instances where GP-PLS-II-U performs significantly better than GP-PLS-II-A in terms of

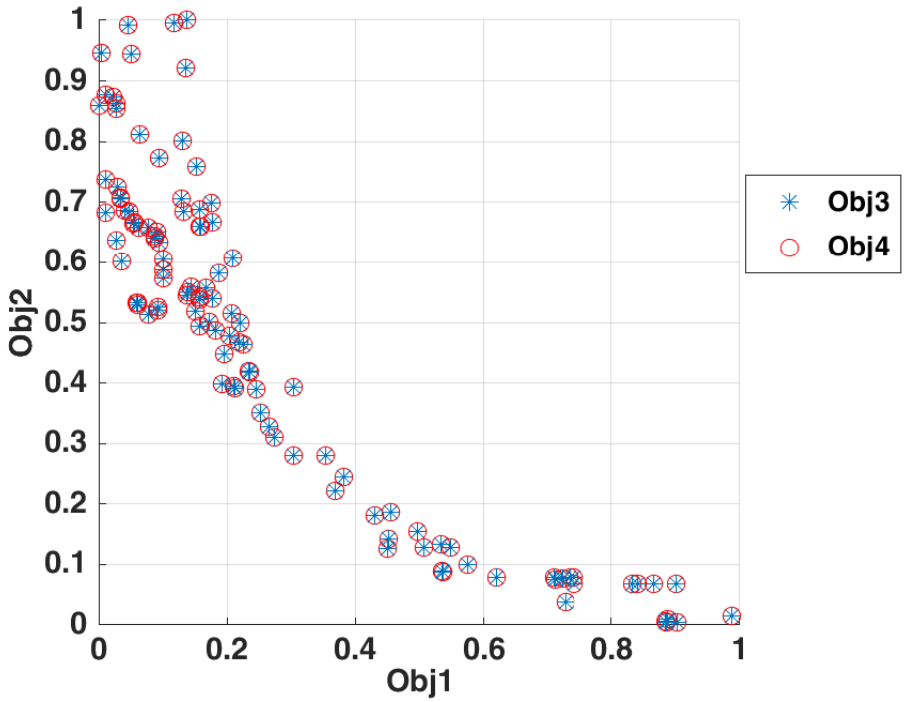
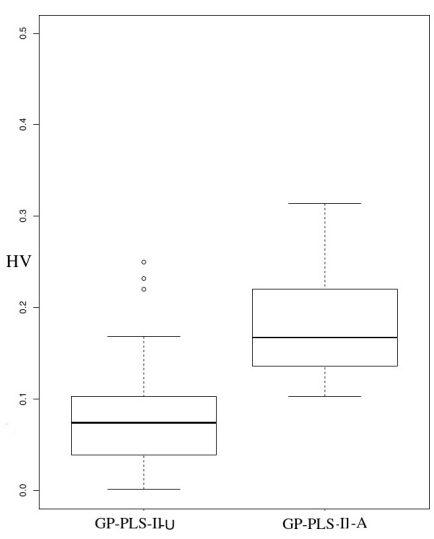


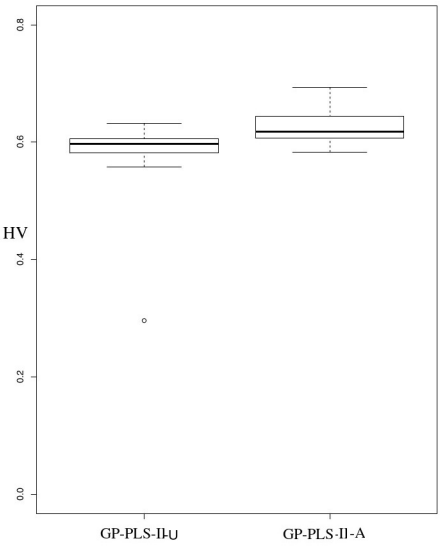
Figure 6.18: Distribution of solutions of GP-PLS-II-A on instance 26.

IGD. From the HV and IGD box plot, we found that generally evolved rules performed much better on the complex test problems. Further, HV and IGD values are more varied on simple problems than the complex problems.

In this study, our experiment results showed that GP-PLS performs much better than the base algorithm, GP-NSGA-III, in terms of both HV and IGD. Moreover, GP-PLS-r performs significantly better than GP-PLS-s. This indicates the effectiveness of using the dominance relation in the comparisons during a local search. Both GP-PLS-II versions performed significantly better than GP-PLS-I-r and GP-PLS-I-s. This result reveals that the selection of solutions based on the FV (convergence and diversity) for neighborhood exploration will improve solutions' quality. GP-PLS-II-A performed significantly better solutions than GP-PLS-II-U and has widely distributed optimal solutions than GP-PLS-II-U.

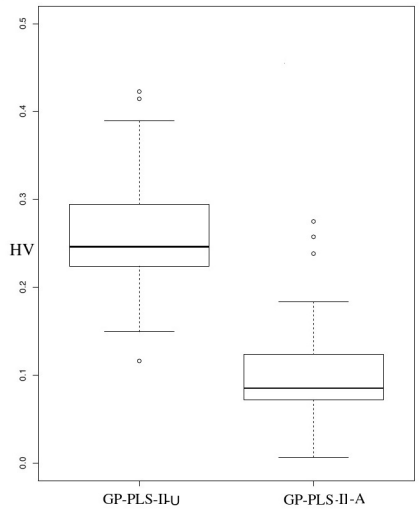


(a) Instance 3.

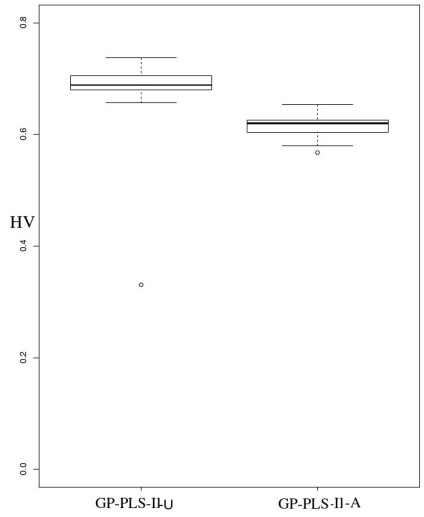


(b) Instance 39.

Figure 6.19: Box-plots of the HV values.

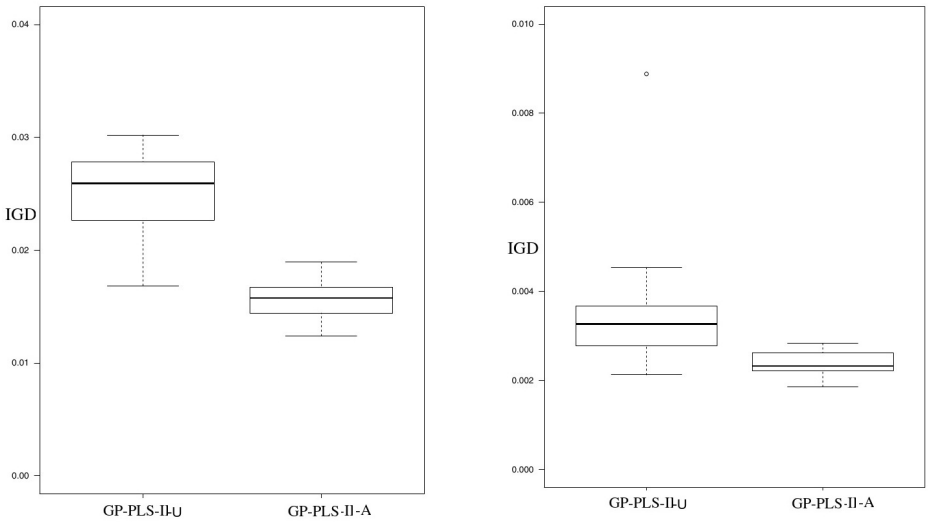


(a) Instance 9.



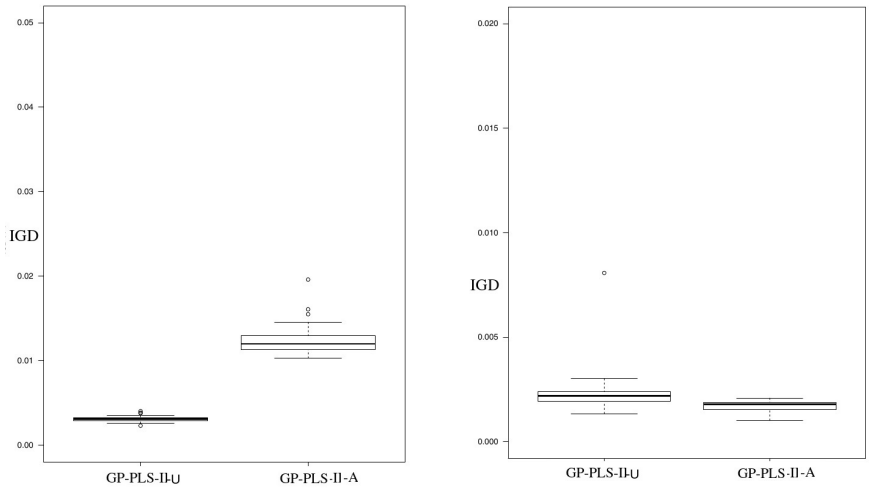
(b) Instance 38.

Figure 6.20: Box-plots of the HV values.



(a) Instance 3. (b) Instance 39.

Figure 6.21: Box-plots of the IGD values.



(a) Instance 9. (b) Instance 38.

Figure 6.22: Box-plots of the IGD values.

6.5 Chapter summary

In this chapter, we combine GP with a PLS for solving many-objective JSS problems. This approach's key idea is to perform multiple local search

steps and effectively find the neighborhood of non-dominated dispatching rules. This local search mechanism helps create excellent exploitation abilities in GP-PLS. GP-PLS features the use of a newly designed restricted neighborhood structure and the partial acceptance mechanism for MaOPs. In this study, two common selection strategies, scalarization, and replacement are experimentally evaluated.From the experiments, we found that the total number of generations and the maximum number of local search steps are highly influential on GP-PLS-I performance. Further, we found that the dominance relation to compare two rules is significantly better than the scalarization strategy.

The second algorithm of GP-PLS-II followed the decomposition-based approach. In this approach, a set of reference points (either uniform or adaptive) decomposed the whole objective space into a number of small subspaces. A fitness-based selection criterion was proposed for selecting initial solutions for neighborhood exploration — the selection criteria based on convergence and diversity.

Extensive experiments have been performed to understand the effectiveness of the proposed GP-PLS as compared to GP-NSGA-III by using the Taillard static job-shop benchmark set. Experiment results showed that GP-PLS performed much better than the GP-NSGA-III algorithm without the local search in terms of both HV and IGD. This study was further analyzed to reveal the different preferences over the use of terminals. This chapter is a first step investigation of PLS in GP. The PLS used in GP improves the effectiveness of evolved rules in terms of HV and GP.

The next chapter discusses the summary of the whole thesis, achieved research objectives, conclusions. It highlights the potential research directions that could be carried out in the future related to the research investigations in this thesis.

Chapter 7

Conclusions

This thesis's overall goal has been successfully achieved, and several new genetic programming-based hyper-heuristics (GP-HH) methods have been developed. These developed methods evolved effective dispatching rules for many-objective job shop scheduling (JSS) problems. The reusability and the effectiveness of the evolved dispatching rules have been demonstrated on the tested benchmark JSS instances. Further, we solved several issues of many-objective JSS problems, such as diversity. This thesis also used Pareto local search (PLS) to improve the quality of evolved dispatching rules.

This chapter sets out the research goals that have been achieved, followed by the main conclusions. Finally, the chapter provides potential research areas of more general future works.

7.1 Achieved objectives

The following research objectives have been fulfilled in this thesis:

- The first research objective developed a new many-objective GP-HH (GP-NSGA-III) to evolve a Pareto-front of non-dominated dispatching rules for JSS. The GP-NSGA-III simultaneously evolved

non-dominated rules for conflicting objectives instead of aggregating many-objective problems into a single-objective optimization problem. GP-NSGA-III seamlessly combined GP-HH for evolving dispatching rules with the selection technique introduced in one of the state-of-the-art-algorithms: NSGA-III. This research objective investigated the trade-offs among many commonly considered objectives in JSS problems, i.e., the mean flowtime (mF) (see equation (2.1) of Chapter 2) , maximal flowtime (maxF) (see equation (2.2) of Chapter 2), mean weighted tardiness (mWT) (see equation (2.6) of Chapter 2) and maximal weighted tardiness (maxWT) (see equation (2.7) of Chapter 2). Furthermore, in this objective, we found that many-objective JSS problems suffered from the same issue of scalability when many-objective JSS problems used multi-objective optimization algorithms (NSGA-II and SPEA2). The results of this research objective in terms of HV and IGD also showed that the proposed algorithm, GP-NSGA-III, is significantly better than the multi-objective optimization algorithms (NSGA-II and SPEA2).

- The second research objective addressed the diversity maintenance issue of many-objective optimization problems. In this objective, we found that uniformly distributed reference points have many useless points [86]. These points fail to find associated Pareto-optimal solutions on the objective space for combinatorial optimization problems. We successfully developed adaptive reference point approaches for generating the reference points according to the candidate solutions' distribution. The following two adaptive reference point approaches were developed: *(i)* model-free approach and *(ii)* model-based approach.

  In the model-free approach (GP-A-NSGA-III(PSO)), particle swarm optimization (PSO) was incorporated into the NSGA-III. Essential changes to particle dynamics in PSO were also introduced in our GP-

A-NSGA-III(PSO) to prevent the majority of reference points from converging to small areas in the objective space. The experiment results showed that GP-A-NSGA-III(PSO) decreased useless reference points during the evolutionary search. Further, the generation of reference points adaptively improved the performance of GP-A-NSGA-III(PSO) in terms of HV and IGD. The results also showed that evolved Pareto-front was more diversified with adaptive reference points than uniform reference points.

In the model-based approach, models were constructed explicitly. They were constructed according to the density of solutions from each defined sub-location in a whole objective space. This *density-based model* learned the distribution of candidate solutions and approximated the Pareto-front based on the evolved solutions. For generating reference points, we introduced two different methods. The first method produced reference points close to vertices, and the second method generated reference points on intermediate locations. Both of these methods enhanced the match between reference points and Pareto-optimal solutions. As a result, useless reference points decreased throughout the evolutionary process. To further improve of the density model, we reduced the density-noise by predicting the mean value in the sub-location.

The model-based approach reduced the useless reference points and provided a better distribution of Pareto-optimal solutions on the entire Pareto-front. Further, a better distribution of reference points improved the diversity of solutions that were observed visually and in terms of HV and IGD. Experimental results demonstrated that the adaptive reference point approach (model-based) performed significantly better than the other compared algorithms on irregular, disconnected, degenerated, and inverted Pareto-front shapes.

- The third research objective combined GP with PLS. The two ver-

sions of GP-PLS algorithms were developed in this research objective, GP-PLS-I and GP-PLS-II. A new fitness-based selection mechanism was developed for the PLS. Both versions of GP-PLS features used a newly designed restricted neighborhood structure and the partial acceptance mechanism for many-objective optimization problems (MaOPs). In this study, we further considered and examined the use of two common selection strategies for the selection of new solutions (scalarization and replacement).

Experimental results demonstrated the effectiveness of the newly developed GP-PLS as compared with the baseline algorithm (GP-NSGA-III). Evolved rules of GP with PLS were more effective than rules evolved by GP without PLS because GP-PLS enhanced the exploitation ability of the algorithm. As a result, rules obtained by GP-PLS had more useful terminals than GP-NSGA-III.

7.2 Main conclusions

The main conclusions for the three research objectives drawn from the three contribution chapters (Chapter 4, Chapter 5, and Chapter 6) are discussed in this section.

7.2.1 Many-Objective GP for JSS

In Chapter 4, we investigated many-objective JSS and focused on evolving a set of trade-offs dispatching rules for many-objective JSS. For this, we combined GP with NSGA-III. In order to tackle many objectives, different aspects have to be considered.

New evolutionary search mechanisms

In this thesis, the evolution process of GP and the selection scheme of NSGA-III were combined. GP is a commonly-used hyper-heuristic for

evolving dispatching rules for JSS and has achieved great success [68, 87]. The combination of GP and NSGA-III was designed as a competitive algorithm for evolving a set of trade-offs rules in many-objective JSS. In this algorithm, GP-NSGA-III combined the initialization, evaluation, and evolutionary operators of GP with the selection scheme of NSGA-III.

Effectiveness of dispatching rules

The experimental results showed that the evolved rules from GP-NSGA-III were significantly better than GP-NSGA-II and GP-SPEA2. Therefore, rules evolved from GP-NSGA-III were well optimized (well-converged and well-diversified). As a result, GP-NSGA-III rules effectively identify and use useful terminals than GP-NSGA-II and GP-SPEA2.

Pareto-front

The experimental results showed that the GP-NSGA-III performed on training instances was significantly better than GP-NSGA-II and GP-SPEA2 in terms of HV and IGD. The evolved rules of GP-NSGA-III are also performed significantly better than GP-NSGA-II and GP-SPEA2 on the test instance, especially on large-size test instances. Thus rules evolved from GP-NSGA-III exhibit the generalization and re-usability abilities.

7.2.2 Non-uniform Pareto-front

In NSGA-III, the diversity of solutions is encouraged by adopting a set of uniformly distributed reference points in the objective space. However, evenly distributed reference points may not be effective for problems with disconnected and non-uniform Pareto-front. These problems generate many useless reference points that are never associated with any of the Pareto-optimal solutions. The existence of these useless reference points in NSGA-III has significant effects on its performance. Two adaptive reference point generation approaches were developed in this thesis

to address the useless reference points issue. These two approaches are described below:

(1) Model-free approach

The first adaptive reference point method inspired by PSO [153] is called "Adaptive-NSGA-III(PSO)". This approach is a simple and efficient one than the model-based approach. We found that useless reference points kept decreased during the evolutionary process. Especially at the later stage of the search, the adaptive reference point scheme led to less useless reference points and a better refinement of the population's regions. The proposed reference point adaptation algorithm significantly improved the performance of GP-NSGA-III in terms of both HV and IGD which indicate that the algorithm performs the best if the distribution of the reference point is consistent with the distribution of the Pareto-optimal solutions.

(2) Model-based approach

In the model-based approach, the models are explicitly constructed and learn the distribution of the candidate solutions of the current generation and generate reference points according to the distribution of the solutions. We also used Gaussian process-based modeling for smoothing the surface and calculated an area under the Gaussian process model's mean function. The model-based approach estimated the approximately accurate number of reference points in each subspace. Furthermore, we found that the model-based approach was effective than the model-free approach. Further, the density-based model is more efficient than Gaussian process-based model. However, Gaussian process-based model is highly effective than the model-free approach and density-based model. The result showed that the number of useless points during the GP search process is constantly decreasing in the model-based reference points adaptation approach.

Benchmarks problems

The performance of the adaptive model-based approach is also verified on ten benchmark problems. These problems have various shapes of Pareto-front. We have used these problems as the testbed in the empirical studies problem, where the number of objectives was scaled from three to eight. The experimental results demonstrated that the model-based approach performed significantly better than the other competing algorithms (REVA*, Two-ARCh, NSGA-III, IMMOEA, and A-NSGA-III) on the different types of Pareto-front which have irregular, disconnected, degenerate, and inverted shapes of Pareto-front. The model-based approach performed significantly better as compared to the other MOEAs with various types of Pareto-front. This is because the model-based approach gets the concrete knowledge of the objective space and provided better matches between the distribution of Pareto-optimal solutions and reference points. Further, a better distribution of reference points improved the diversity of the solutions in terms of HV and IGD.

7.2.3 Pareto local search (PLS)

In Chapter 6, we combined GP with PLS. The first GP-PLS mechanism, GP-PLS-I investigated the influence of Pareto local search with GP. Based on the results of GP-PLS-I, a new algorithm, GP-PLS-II, was developed. These findings are as follows: (1) GP-PLS required a fitness-based selection mechanism for selecting the initial solution for neighborhood exploration. (2) GP-PLS should perform an adequate number of local search steps for local searches to show its effectiveness, and (3) The total number of generations and the maximum number of local search steps significantly affected by the performance of GP-PLS. Both GP-PLS algorithms used the three major components of PLS, (1) selection of a solution, (2) neighborhood exploration, and (3) comparison. From the GP-PLS, we found that the fitness-based selection mechanism based on the convergence and di-

versity criteria for selecting initial solutions performed significantly better than the random selection. Further, a neighborhood structure for GP, the restricted mutation, can effectively prevent a new neighboring rule discovered during the local search process from being significantly different from the original rule. For comparing the new rule with its immediate parent rule, the following two strategies were considered: (1) the scalarization strategy [79], and (2) the replacement strategy [26]. The result showed in terms of HV and IGD indicates the effectiveness of using the dominance relation in the comparisons during the local search.

Extensive experiments were performed to understand the effectiveness of the proposed GP-PLS as compared with the base-line algorithm, GP-NSGA-III. Experiment results showed that GP-PLS performs much better than the current state-of-the-art method without local search in terms of both HV and IGD. Moreover, programs obtained by GP-PLS-II and GP-PLS-I were more effective than the GP-NSGA-III because GP-PLS enhances the exploitation ability; as a result, GP-PLS algorithms have the majority of useful terminals.

Sensitivity analysis

In the experiment, a sensitivity analysis was performed and found that the total number of generations and the maximum number of local search steps together provide varied trade-offs between global and local searches in GP-PLS.

7.3 Future work

This section highlights key areas of future work related to the field of many-objective JSS.

7.3.1 Incorporate user preferences to many-objective JSS

This thesis used reference points that decomposed the original many-objective space into subspace. However, we can further use these reference points to target the user's interested in different sub-regions of the entire Pareto-front. The use of user preferences is important for MaOPs because the user may have a preferred subset of Pareto-optimal solutions that fall specific subregion of the objective space. Also, it has been shown in the literature that reference points are one of the efficient and effective methods to preference articulation [31]. This preference articulation is particularly crucial in many-objective where it is improbable to cover the whole Pareto-front [31] comprehensively.

7.3.2 Incorporate locality of search operators to Genetic Programming

As we know, every optimization problem can be decomposed into a genotype-phenotype mapping and a phenotype-fitness mapping. In EC, locality refers to how well neighboring genotypes correspond to neighboring phenotypes. The locality of a representation is high if all neighboring genotypes correspond to neighboring phenotypes. In contrast, the locality of a representation is low if some neighboring genotypes do not correspond to neighboring phenotypes. It is already mentioned that a representation of high locality is necessary for efficient evolutionary search [52, 171]. In most local search methods, mutation use to explore the neighborhood of its immediate parents. If the mutation-based search algorithm does not have a high locality representation, it would jump randomly around the search space.

There is no explicit genotype-phenotype mapping in the tree-structured GP, so we can say that there are no explicit phenotypes distinct from genotypes [52]. This is common in GP, therefore, to study instead of the behavior of the mapping from genotype to fitness. However, in JSS, a

rule is represented as a GP tree and two dispatching rules with different tree structures can essentially have exactly the same behavior in making decisions [130]. For example, the dispatching rules with the priority functions of x, $2x$ will always make the same decision [66]. So, in JSS, rules are different in terms of phenotypic behavior instead of genotypic structure. So to compare the similarity of solutions during neighborhood exploration in PLS, we will adopt a phenotypic characterization approach [66] which can characterize the behavior of a dispatching rule as a fixed-length numeric vector. This phenotypic characterization approach is based on a reference dispatching rule and applied to single objective JSS problems. However, it is a potential research direction for many-objective JSS. Therefore, we will explore this research direction in the future and find reference rules for many-objective JSS. Moreover, we will also integrate this approach with our PLS algorithm (see Chapter 6).

7.3.3 Incorporate effective crossover operator to many-objective JSS

This thesis tackles several issues of many-objective JSS. One of the challenges of many-objective JSS problems is designing an effective crossover operator. Sato et al. [174] revealed that in combinatorial optimization problems such as JSS problems, the recombination of two parents close to the Pareto-front might generate an offspring distant from the Pareto-front since a conventional crossover operator might be too disruptive for many-objective combinatorial optimization. This is because, in many-objective JSS problems, non-dominated solutions can have quite different building blocks, which are useful for different objectives. Randomly selecting parents based on the dominance relation may result in combining building blocks for different objectives together which may not lead to good offspring. So, in this case, a good recombination operator is needed to diminish crossover operators' disruptive effect in many-objective JSS problems.

7.3.4 Incorporate adaptive terminal selection to many-objective JSS

This thesis developed several GP-HH approaches for many-objective JSS. Each approach used a GP system for evolving and effective rules. These rules utilized several potential terminals (and functions) that can be given as parameters to the GP system to discover dispatching rules automatically. The terminal set was formed by machine, job, and shop attribute [47, 150]. Although GP based approaches in this thesis outperformed the other compared algorithms, the terminal selection was not based on the preference objectives. The size of dispatching rules also increases with the number of objectives and rules may have redundant terminals that affect the interpretability of dispatching rules. The interpretability of rules can be achieved by eliminating the irrelevant terminals from the terminal set.

7.3.5 Dispatching rules for many-objective dynamic JSS

In this thesis, we have only used the GP to evolve dispatching rules (priority functions) for static JSS environments because many-objective research is comparatively newer than single objective JSS and multi-objective JSS. Furthermore, dynamic JSS environments are more complex than static JSS environments. This is because, in dynamic JSS environments, jobs continuously arrive on the shop floor at various instances of time with no prior process information, e.g., release date, due date, and processing time are not given before job arrival. Further, the existing GP-HH for dynamic JSS is not efficient regarding fitness evaluation. Therefore, this thesis evolved effective and efficient dispatching rules for static JSS. This thesis assisted in the understanding of how optimization approaches can be adapted to improve many-objective JSS algorithms performances. Therefore, there is a need to investigate dynamic many-objective JSS and evolve effective dispatching rules.

7.3.6 Dispatching rules for many-objective flexible JSS

Flexible job-shop scheduling (FJSS) problems [16] is an extension of classical JSS problems that allows the processing of each operation on more than one machine. FJSS problems consist of two sub-problems: (1) the routing sub-problem and (2) the scheduling sub-problem. The routing sub-problem focuses on assigning each operation to a machine out of the set of capable machines. The scheduling sub-problem aims to sequence the assigned operation on all the machines to obtain a feasible schedule that optimizes one or many objectives. It is more complicated than JSS problems because it involves two strongly coupled decisions, the routing sub-problem and the scheduling sub-problem [16]. FJSS is an NP-hard problem[16]; therefore, it is essential to know how to efficiently use the GP system and evolve effective rules for many-objective FJSS problems.

Furthermore, over the past decades, the single-objective FJSS problems mainly focused on minimizing the makespan that has been extensively studied in the JSS literature. Compared to the single-objective FJSS problems, the multi-objective FJSS problems research is relatively limited to three objectives [202]. However, many real-world scheduling problems involve the simultaneous optimization of several objectives that conflict. Despite the requirement of many-objective research, there are only a few studies on many-objective FJSS [57] but in these studies, many-objective optimization was not explicitly investigated [57]. In line with this research trend, we believe that many-objective FJSS will attract increasing research attention in the near future. Therefore, research to construct a hyper-heuristic framework to generate effective rules for many-objective FJSS problem is a potentially significant research direction that demands a new and specialized GP approach.

In summary, many future research directions arise as a result of the work discussed in this thesis.

7.4 General Considerations

This section first describes the main component of the proposed algorithms with algorithmic parameters. Next, this section shows the application of the proposed algorithm to a cloud platform.

7.4.1 Main components of thr Proposed algorithm

This section shows the main components and characteristics of the developed algorithm. Figure 7.1 shows these three components of the hybridized algorithm developed in this thesis.

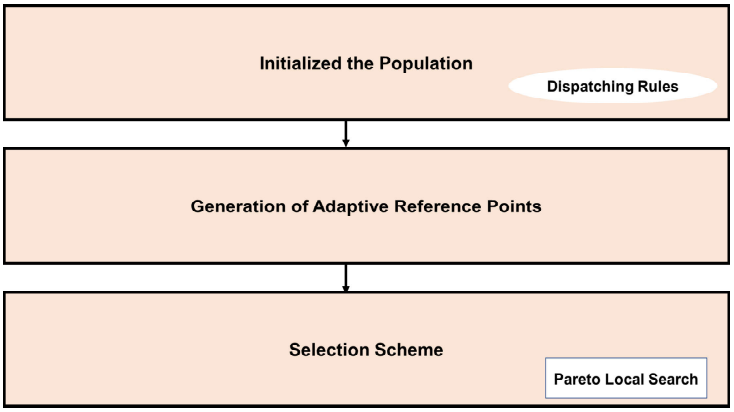


Figure 7.1: Main Components of Algorithm

Initialized the Population

In this thesis, we used the GP system to evolve automated dispatching rules for JSS problems represented as trees and made up of function and terminal nodes. Terminals nodes are usually defined by constants and variables from the specific problem domains.

The GP system usually requires genetic parameters such as the population size, termination parameter, crossover and mutation probabilities, and their associated parameters. These parameters are not algorithmic parameters but problem-specific parameters to generate a population of individuals.

Although we evolved dispatching rules for static JSS problems, dispatching rules can cope in dynamic environments, such as cloud scheduling [188] and vehicle routing problems [16]. Dispatching rules is one of the potential approaches to deal with dynamic changes because they are computationally efficient and can react quickly to dynamic changes on the shop floor [141, 159]. So, one can also use GP based system for evolving dispatching rules for dynamic problems.

Our algorithm is not limited to generate dispatching rules by using GP system but one can generate a population of individuals from any other benchmark functions (see Appendix A) and feed these individuals to the next component of algorithms to select the best solution.

Reference point adaption method

In real-world scheduling problems, the true Pareto-front is unknown and due to the discrete and combinatorial solution space, the Pareto- front of scheduling problems can be very irregular. It was shown in this thesis that the adoption of uniformly distributed reference points affects the performance of the algorithms adversely. Therefore, we developed three adaptive reference points approaches (PSO-based adaptive reference points approach, density-based model adaptive reference points approach, and gaussian process model adaptive reference points approach).

The PSO-based adaptive reference point approach is more efficient than the other two approaches but it requires PSO parameters such as inertia weight and acceleration coefficients [101].

The density-based model, adaptive reference points approach, is more effective than the PSO-based model adaptive reference and more efficient

than the Gaussian process model adaptive reference points approach. This approach does not require any controlled parameters.

The Gaussian process model, adaptive reference points approach, is more effective than the other two approaches. This approach requires the hyperparameters of the covariance functions [170].

These adaptive reference points approaches which are mentioned above require the size of reference points but the number of reference points is not an algorithmic parameter, as this is entirely at the disposal of the user. The population size is dependent on the size of reference points, as population size $\approx$ number of reference points.

Further, these adaptive reference point methods might be easily deployed with any other reference points-based evolutionary multi-objective optimization (EMO) algorithms to improve their performance on problems with irregular Pareto-front.

Selection scheme

The algorithm in this thesis solves many-objective JSS problems by combining GP and selection schemes of MaOPs. The new fitness-based selection mechanism based on convergence and diversity was introduced. This new selection scheme balance the convergence criteria and diversity criteria and does not require to set any new parameter.

To improve the effectiveness of the solution, we further integrate global search with the PLS. Here, PLS requires two additional parameters: the archive's size and the maximum number of local search steps.

7.4.2 Cloud task scheduling problem

In this section, we apply our many-objective scheduling algorithm on the cloud task scheduling problem. However, we already applied our algorithm to the number of benchmark test problems (see Appendix A)

with three to eight objectives to show our many-objective scheduling algorithm's general applicability and usefulness.

In cloud computing, users may utilize hundreds or thousands of virtualized resources and it is impossible for everyone to allocate each task manually. Task scheduling problem refers to how to reasonably arrange many tasks provided by users in virtual machines, which is very important in cloud computing.

In cloud computing, the cost, response time, resource utilization, and load balancing are often conflicting [54]. Hence cloud scheduling can be considered as a many-objective optimization problem. To achieve better scheduling performance, one should to minimize cost, minimize time, improve resource utilization, and load.

In order to test the performance of the algorithm on the task scheduling problem, we can use discrete event simulation (DES) platforms [22] to support cloud computing environments such as the Cloudsim platform [22]. First of all, detailed parameter settings of the virtual machine and tasks should be provided. Also, initialize the population of size N. The GP system can randomly initialize the population, where each individual represents a dispatching rule. After objective values are calculated by applying the rule to the training instances of cloud scheduling. The fitness of dispatching rules is evaluated by a DES. Then the following steps can use for the selection of N best solutions

1. combine offspring with the parent population and obtain a combined population of size 2N.

2. generate reference points adaptively which split the objective space into a number of independent sub-regions. This partitions the population into multiple subpopulations by associating each individual with its closest reference points.

3. Assign fitness values using equation (7.1) (where d_1 represents the distance from solution to the ideal point. Similarly, d_2 represents the

inverse of the acute angle between solutions and reference points) to each solution and N solutions are selected based on their fitness values.

$$FV = d1 + \frac{d2}{\theta_m}. \tag{7.1}$$

This method not only makes a greater contribution to the convergence performance of the entire population but also improves the diversity of the population. For evaluating the performance of the algorithm on the cloud scheduling problem, we perform simulation experiments using the Cloudsim platform.

Appendix A

Further studies

A.1 Introduction

Although the motivation of our research focuses on JSS problems, to show the general applicability, usefulness, and effectiveness of the MARP-NSGA-III (see Chapter 5). We also apply MARP-NSGA-III to the benchmark test problems to further check the effectiveness of the MARP-NSGA-III. In the benchmark problem case, we have compared the performance of MARP-NSGA-III with NSGA-III [43] and A-NSGA-III [85] with three, five, and eight objectives.

A-NSGA-III is an updated version of NSGA-III, where RVEA* is an extension of RVEA [31]. These extensions are mainly relocating reference points which were suggested to handle problems with irregular Pareto-front. IMMOEA [30] is a model-based method that constructs Gaussian process-based inverse models that map all the non-dominated solutions from the objective space to the decision space.

In these experiments, we focus mainly on ten benchmark problems with irregular Pareto-front. These problems are *DTLZ4, DTLZ5, and DTLZ7* [31], *Inverted DTLZ1 (IDTLZ1), Inverted DTLZ2 (IDTLZ2)* [76], *WFG1, WFG2, WFG9* [31], *MAF1*,and *MAF2* [92]. The characteristics of all the ten benchmark problems are summarized in Table A.1 of the next

Table A.1: The characteristics of benchmark problems

Problems	Characteristics
IDTLZ1	inverted
IDTLZ2	Convex, inverted
DTLZ4	Concave,biased
DTLZ5	Concave, degenerate
DTLZ7	Mixed, disconnected, multi-model
MAF1	Linear, multi-model, inverted
MAF2	Disconnected
WFG1	Sharp tails
WFG2	Disconnected
WFG9	Concave

section.

A.2 Benchmark functions on MaOPs

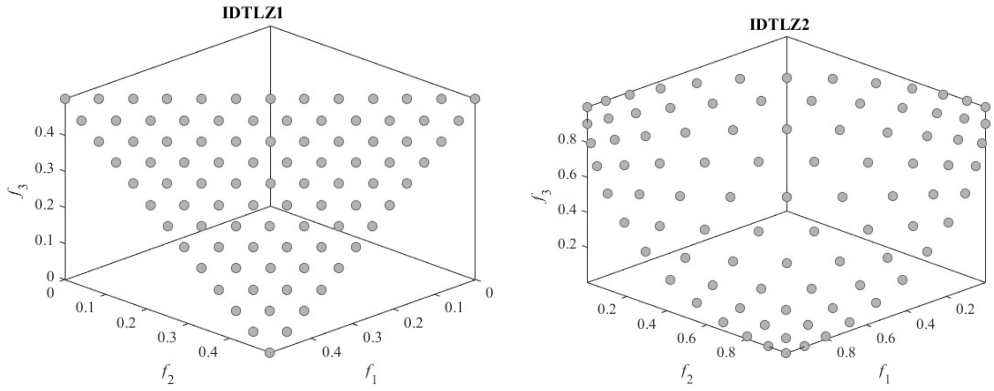
Benchmark function are widely used in the literature to evaluate MaOPs algorithms' performance on problems with irregular Pareto-front.

In this thesis, we used Inverted DTLZ1 (IDTLZ1), Inverted DTLZ2 (IDTLZ2), *DTLZ4*, *DTLZ5*, *DTLZ7*,*WFG1*, *WFG2*, *WFG9*, *MAF1*, and *MAF2* benchmark problems. The Pareto-front for these problems is also irregular. *MAF1* and *MAF2* are taken from the *CEC-2017* competition on evolutionary many-objective optimization [32]. The characteristics of all the 10 benchmark problems are summarized in Table A.1. It should be noted here that all the figures of benchmark problems are taken from MATLAB-based EMO platform called PlatEMO [187].

A.2.1 IDTLZ1 and IDTLZ2 problems

IDTLZ1 and IDTLZ2 denote the problems of inverted DTLZ1 and DTLZ2, respectively, where the regular Pareto-front of DTLZ1 and DTLZ2 are in-

verted and thus become irregular [86]. This test problem is used to assess whether MOEA is capable of dealing with an inverted Pareto-front. Figures A.1 (a) and A.1 (b) shows IDTLZ1 and IDTLZ2, respectively.



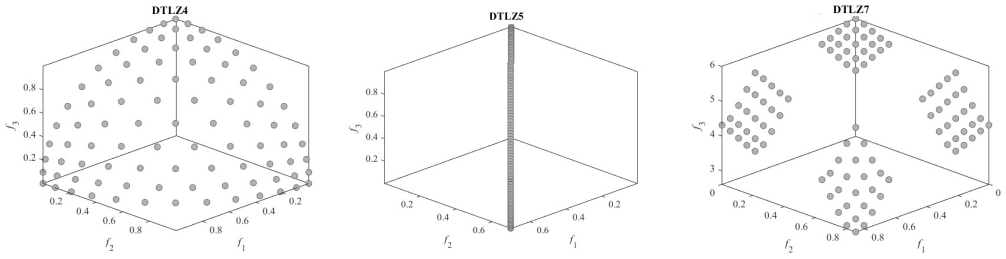
(a) IDTLZ1.

(b) IDTLZ2.

Figure A.1: The Pareto-front with three objectives on IDTLZ problems.

A.2.2 DTLZ-4, DTLZ5, and IDTLZ-7 Problems

The Pareto-front of DTLZ-4 has a badly-scaled Pareto-front (where each objective function is scaled to a substantially different range), especially when the fitness landscape is highly multi-modal. DTLZ-5 is originally intended to be test problems with degenerated Pareto-front. The shape of the true Pareto-front is a 1-D curve which shows this problem independent of the number of objectives. The shape of the true Pareto-front of DTLZ-7 is disconnected. This problem is usually used to assess the capability of those MOEAs which have disconnected Pareto-front. Figures A.2 (a), A.2 (b), and A.2 (c) show DTLZ4, DTLZ5, and DTLZ7, respectively.



(a) DTLZ4. (b) DTLZ5. (c) DTLZ7.

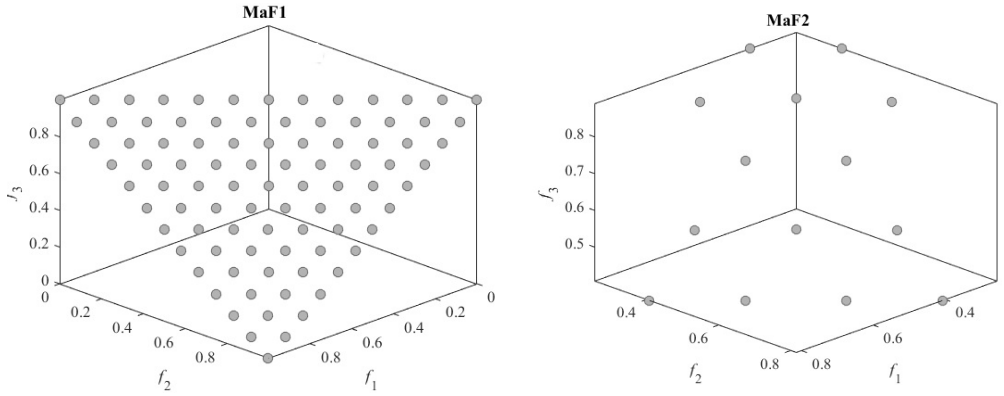
Figure A.2: The Pareto-front with three objectives on DTLZ problems.

A.2.3 MAF1 and MAF2 Problems

MAF1 is a modified version of IDTLZ1. The feasible range of MAF1 is [0,1] on the objective space. This test problem is used to assess whether MOEAs are capable of dealing with inverted Pareto-front. MAF2 increases the difficulty of convergence in DTLZ2. In DTLZ2, all in the do not have to be optimized simultaneously in order to reach the true Pareto-front. In contrast, the MAF2 problem is used to assess whether the MOEAs can perform concurrent convergence on different objectives. Figures A.3 (a) and A.3 (b) show MAF1 and MAF2, respectively.

A.2.4 WFG1,WFG2 and WFG9 Problems

The WFG test problems (WFG1, WFG2, and WFG9) have scaled true Pareto-front. These WFG test problems are widely used to test the performance of MOEAs on MaOPs [31]. WFG1 contains both convex and concave segments. Whereas the WFG2 test problem is capable of dealing with scaled disconnected Pareto-front. WFG9 has concave Pareto-front. Moreover, its fitness landscape is highly multi-modal. Figures A.4 (a), A.4 (b), and A.4 (c) show WFG1, WFG2, and WFG9, respectively.



(a) MAF1. (b) MAF2.

Figure A.3: The Pareto-front with three objectives on MAF problems.

A.3 Experiment design

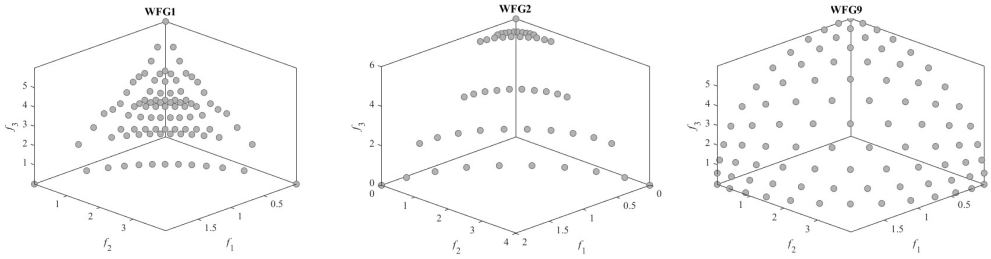
A.3.1 Parameter setting of benchmark problems

The number of decision variables for DTLZ and inverted DTLZ test problems is set as recommended in [45]. The MAF problems, the solution dimension is set according to [92], whereas, for WFG problems, we follow the recommendation of decision variables in [31]. The population sizes

Table A.2: Number of Reference Points and Population Size for DTLZ and MAF.

No. of Obj(M)	Ref. Points(H)	Pop. Size(N)
3	91	92
5	210	212
8	156	156

for DTLZ and MAF problems of all compared algorithms are shown in **Table** A.2. The settings for WFG are shown in **Table** A.3. For DTLZ and MAF problems, the maximum number of generations is adopted from [43]



(a) WFG1. (b) WFG2. (c) WFG9.

Figure A.4: The Pareto-front with three objectives on WFG problems.

Table A.3: Number of Reference Points and Population Size for WFG.

No. of Obj(M)	Ref. Points(H)	Pop. Size(N)
3	105	108
5	175	176
8	217	220

and [31] respectively. The parameter setting of WFG is taken from [31].

A.3.2 Algorithms parameter settings

The widely used genetic operators, i.e., the simulated binary crossover (SBX) [43] and the polynomial mutation [42] are employed to create the offspring population, as in many other MOEAs [42, 43]. The evolution operator is kept identical for all approaches: (i) SBX probability (p_c) is set to 1.0, (ii) polynomial mutation probability (p_m) is set to 1/n (where n is the number of variables) [43]. The distribution index(η_c) is kept as 30 for SBX crossover and polynomial distribution index (η_m) are set to 20. For RVEA*, the penalty parameter α is set to 2, and the frequency of reference point adaptation f_r is set to 0.1.

A.4 Results and discussion

In the experiment, 30 independent runs are carried out for each algorithm. Then, the mean and standard deviations of HV and IGD values were reported. The best value for each problem was marked in boldface. The benchmark problems are substantially studied in the literature. The source codes of RVEA* [31], Two-ARCh [167], and IMMOEA [30] are available on different platforms. Therefore, we compare these algorithms with our proposed algorithms (MARP-NSGA-III) with three, five, and eight objectives.

A.4.1 Performance of obtained solutions

Tables A.4 and A.5 present the HV and IGD values scored by the proposed algorithm and other compared algorithms. In general, MARP-NSGA-III has achieved significantly better performance on 15 of 30 instances in HV and 14 of 30 instances in IGD. On the other hand, RVEA* is a very competitive algorithm and has the best performance in nine instances.

A.4.2 Further analysis

Performance on IDTLZ1 and IDTLZ2

IDTLZ1 has an inverted Pareto-front and many uniformly distributed reference points created on the normalized hyper-plane will not have an associated Pareto-optimal point. Therefore, adaptive reference points play an important role in achieving a higher degree of population diversity than uniformly distributed reference points. Tables A.4 and A.5 show that A-NSGA-III, RVEA*, and MARP-NSGA-III have better HV and IGD values because they can generate higher population diversity than the NSGA-III with predefined uniformly distributed reference points. The statistical results in Tables A.4 and A.5 reveal that the MARP-NSGA-III performs significantly better than the other methods in the five-objective instances. However, it has a competitive performance in eight-objective cases. In the

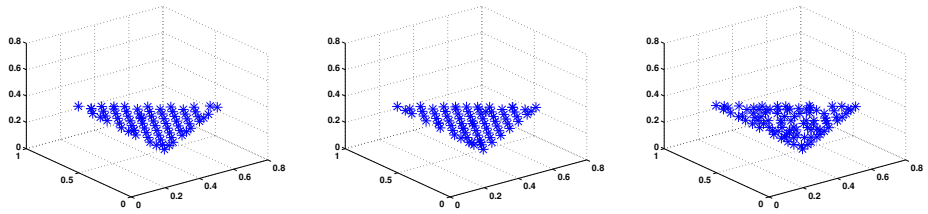
Table A.4: The mean and standard deviation ($\bar{x} \pm \sigma$) over the average HV values on M-objectives on IDTLZ1, IDTLZ2, DTLZ4,DTLZ5, DTLZ7, MAF1, MAF2, WFG1, WFG2 problems.

HV $Mean(std)$							
Function	M	NSGA-III	A-NSGA-III	RVEA*	Two-ARCh	IMMOEA	MARP-NSGA3
Inv-DTLZ1	3	1.07e-1(4.0e-3)	1.30e-1(2.8e-3)	1.32e-1(3.9e-3)	1.19e-1(1.5e-3)	1.29e-1(3.00e-3)	1.33e-1(3.4e-3)
	5	3.91e-3(4.2e-4)	4.2e-3(4.9e-4)	4.23e-3(3.4e-5)	4.32e-3(8.4e-6)	4.34e-3(1.5e-4)	**5.3e-3(6.7e-4)**
	8	6.22e-4(3.8e-5)	**7.05e-4(1.5e-3)**	6.66e-4(8.8e-5)	5.81e-4(3.7e-3)	5.71e-4(1.9e-5)	6.67e-4(4.0e-4)
Inv-DTLZ2	3	4.52e-1(3.1e-3)	4.72e-1(3.0e-3)	4.58e-1(9.5e-4)	4.857e-2(7.0e-3)	**5.26e-1(8.8e-3)**	4.62e-1(3.7e-3)
	5	6.30e-2(2.4e-3)	6.89e-2(5.9e-3)	7.00e-2(2.9e-3)	9.94e-2(2.8e-3)	8.60e-2(5.5e-3)	**1.07e-1(6.9e-3)**
	8	6.62e-3(6.8e-4)	9.37e-3(1.1e-4)	2.7e-3(4.3e-4)	1.47e-1(4.9e-4)	2.577e-3(3.7e-4)	**1.54e-2(1.7e-4)**
DTLZ4	3	7.02e-1(1.2e-2)	7.47e-1(3.1e-3)	7.34e-1(1.1e-2)	7.19e-1(4.1e-2)	6.97e-1(5.1e-2)	**7.66e-1(5.5e-4)**
	5	7.03e-1(2.6e-3)	7.17e-1(1.2e-4)	7.40e-1(4.2e-4)	7.32e-1(3.7e-3)	7.20e-(5.2e-4)	**7.59e-1(3.7e-3)**
	8	8.45e-1(2.6e-4)	8.59e-1(5.5e-3)	8.71e-1(1.6e-4)	8.67e-1(9.7e-4)	8.57e-1(1.7e-4)	**8.74e-1(9.4e-4)**
DTLZ5	3	8.19e-2(1.7e-2)	8.55e-2(6.3e-4)	9.19e-2(2.9e-3)	**1.3e-1(6.1e-4)**	8.23e-2(1.4e-3)	8.49e-2(7.2e-4)
	5	5.28e-1(2.6e-2)	6.9e-1(3.7e-2)	7.48e-1(2.8e-4)	6.6e-3(1.6e-4)	**7.87e-1(2.2e-4)**	5.3e-1(4.4e-2)
	8	6.03e-1(1.9e-2)	6.40e-1(2.3e-2)	5.62e-1(1.7e-4)	6.37e-1(3.8e-6	5.56e-1(4.99e-2))	6.38e-1(2.1e-2)
DTLZ7	3	3.25e-1(7.9e-3)	3.48e-1(9.4e-3)	3.44e-1(8.3e-3	3.53e-1(5.8e-2)	3.14e-1(9.6e-2)	3.58e-1(1.2e-3)
	5	2.240e-1(3.8e-3)	3.23e-1(6.3e-3)	2.970e-1(2.1e-2)	1.35e-1(2.3e-2)	3.21e-1(1.8e-3)	**3.5e-1(7.2e-2)**
	8	3.08e-1(4.8e-3)	3.25e-1(6.3e-3)	2.98e-1(1.7e-3)	1.56e-1(4.9e-4)	2.56e-1(4.99e-2)	**3.31e-1(2.3e-2)**
MAF1	3	1.15e-1(2.0e-3)	1.34e-1(3.0e-4)	**2.88e-1 (1.0e-3)**	2.20e-1 (6.59e-3)	2.80e-1 (7.0e-4)	1.38e-2(3.4e-3)
	5	2.04e-2(2.9e-4)	2.62-2(3.8e-4)	6.35e-2 (7.93e-4)	2.04e-2(3.01e-4)	7.23e-3 (5.50e-4)	**7.18e-2(4.77e-3)**
	8	6.57e-4(1.0e-5)	6.24-4(6.9e-5)	6.69e-4(4.9e-5)	6.60e-4(4.3e-5)	6.44e-4(3.9e-4)	**6.75e-4(4.7e-5)**
MAF2	3	3.30e-1(2.1e-2)	4.05e-1(3.9e-3)	**4.19e-1(1.4e-3)**	3.66e-1(3.0e-3)	3.48e-1(2.2e-3)	3.94e-1(2.3e-3)
	5	1.22e-1(1.1e-2)	1.57e-1(1.3e-2)	1.61e-1(2.3e-2)	1.52e-1(2.1e-2)	1.38e-1(4.3e-2)	**1.68e-1(5.3e-2)**
	8	1.77e-1(1.82e-3)	2.03e-1(3.2e-3)	1.60e-1(2.4e-3)	1.57e-1(9.7e-3)	1.32e-1(6.6e-3)	**2.30e-1(4.2e-3)**
WFG1	3	8.48e-1(5.2e-2)	8.44e-1(5.0e-2)	**8.70e-1 (3.3e-2)**	8.56e-1 (3.9e-2)	8.11e-1 (2.74e-2)	8.22e-1(8.4e-2)
	5	7.09e-1(1.6e-2)	**8.39e-1(1.9e-2)**	8.33e-1(3.4e-2)	8.32e-1(5.1e-2)	8.06e-1(2.6e-2)	8.29e-1(1.9e-2)
	8	6.09e-1(4.4e-2)	8.35e-1(5.9e-2)	8.29e-1(7.4e-2)	8.34e-1(3.3e-2)	7.86e-1(4.6e-2)	**8.38e-1(4.9e-2)**
WFG2	3	7.07e-1(4.99e-3)	8.44e-1(8.3e-2)	8.55e-1 (4.3e-2)	7.61e-1 (4.5e-2)	7.70e-1 (2.74e-2)	8.42e-1(7.0e-2)
	5	9.89e-1(2.2e-3)	9.53e-1(6.4e-2)	9.86e-1(2.6e-2)	9.80e-1(5.6e-2)	9.36e-1(3.6e-2)	**9.89e-1(3.3e-2)**
	8	9.22e-1(9.5e-2)	9.40e-1(7.5e-2)	**9.52e-1(8.6e-2)**	9.46e-1(7.6e-2)	9.17e-1(7.2e-2)	9.41e-1(6.0e-2)
WFG9	3	6.20e-1(4.8e-2)	**6.75e-1(4.9e-2)**	6.64e-1(1.4e-2)	6.66e-1(3.0e-2)	6.48e-1(3.0e-2)	6.68e-1(3.9e-2)
	5	6.22e-1(6.5e-2)	6.52e-1(5.6e-2)	**7.0e-1(5.0e-2)**	6.69e-1(2.1e-2)	6.38e-1(5.7e-2)	6.53e-1(5.5e-2)
	8	7.19e-1(6.9e-2)	7.28e-1(3.2e-2)	8.28e-1(1.9e-2)	7.77e-1(1.7e-2)	7.07e-1(2.7e-2)	**8.31e-1(4.9e-2)**

Table A.5: The mean and standard deviation ($\bar{x} \pm \sigma$) over the average IGD values on M-objectives on IDTLZ1, IDTLZ2, DTLZ4, DTLZ5, and DTLZ7, MAF1, MAF2, WFG1, WFG2 problems.

IGD $Mean(std)$							
Function	M	NSGAIII	A-NSGA-III	RVEA*	Two-ARCh	IMMOEA	MARP-NSGAIII
IDTLZ1	3	3.58E-2(1.0e-3)	2.47e-2(8.0e-3)	2.37-02(2.7e-2)	1.66e-1(1.6e-2)	4.22e-1(1.0e-3)	2.38e-2(2.1e-3)
	5	8.62e-2(3.8e-3)	5.53e-2(8.3e-3)	1.47e-1(2.0e-2)	1.90e-1(3.2e-2)	6.75e-1 (4.36e-1)	**3.13e-2(8.3e-3)**
	8	9.62e-2(8.8e-3)	**5.13e-2(9.2e-3)**	2.45e-1(2.5e-2)	1.96e-1(2.0e-2)	4.85e-1(3.0e-2)	5.49e-2(8.9e-3)
IDTLZ2	3	6.73e-2(8.3e-3)	6.42e-2(5.6e-3)	8.07e-2(1.0e-3)	9.0e-2(7.0e-3)	**6.20e-2(4.0e-3)**	6.26e-2(2.5e-3)
	5	2.33e-1(1.2e-2)	2.03e-1(1.2e-2)	2.73e-1(1.3e-3)	2.1e-1(5.8e-3)	3.36e-1 (1.2e-2)	1.78e-1(1.8e-2)
	8	5.62e-1(3.8e-2)	5.16e-1(2.3e-2)	6.10e-1(6.5e-3)	4.1e-1(6.7e-3)	5.44e-1(1.5e-2)	**3.11e-1(2.3e-2)**
DTLZ4	3	7.05e-2(6.01e-3)	6.26e-2(6.2e-3)	7.07e-2(8.4e-2)	9.06e-2(1.4e-2)	7.57e-2(2.6e-3)	**6.2e-2(6.43e-3)**
	5	1.63e-02(4.3e-4)	1.64e-2(5.8e-4)	1.53e-2(5.4e-4)	1.71e-2(1.4e-4)	1.68e-2(3.9e-4)	**1.39e-2(3.9e-4)**
	8	2.68e-02(1.0e-3)	**4.07e-3(5.7e-4)**	3.25e-2(2.63e-3)	2.28e-2(1.33e-3)	1.22e-1(1.03e-3)	2.16e-2(1.9e-3)
DTLZ5	3	2.18e-2(2.2e-3)	1.20e-2(1.7e-3)	1.37e-2(1.0e-3)	1.22e-2(7.0e-3)	1.96e-2(4.0e-3)	1.75e-2(1.4e-3)
	5	2.70e-1(5.16e-2)	1.99e-1(5.7e-2)	2.48e-1(2.8e-4)	2.6e-3(1.6e-4)	**1.87e-1(2.2e-4)**	2.26e-1(3.9e-2)
	8	4.04e-1(8.8-e-2)	4.01e-1(7.2e-2)	3.62e-1(1.7e-4)	6.377e-1(3.8e-6)	5.56e-1(4.99e-2))	**4.3e-1(9.1e-2)**
DTLZ7	3	9.44e-2(4.99e-3)	6.79e-2(5.2e-3)	7.05e-2(1.56e-3)	7.15e-2(8.5e-2)	8.34e-2(3.9e-2)	6.75e-2(3.03e-3)
	5	4.54e-1(2.2e-2)	3.44e-1(2.4e-2)	4.98e-1(9.0e-3)	5.3e-1(9.0e-2)	5.65e-1 (2.74e-2)	**3.08e-1(4.1e-2)**
	8	7.89e-1(3.7e-2)	7.61e-1(2.7e-2)	7.82e-1(1.6e-1)	7.53e-1(8.6e-2)	8.14e-1(9.6e-2)	**7.44e-1(8.6e-2)**
MAF1	3	6.44e-2(2.3e-3)	5.07e-2(1.1e-3)	4.59e-2 (9.4e-4)	**4.37e-2 (4.3e-4)**	1.09e-1 (9.9e-3)	4.67e-2(3.6e-3)
	5	1.72e-1(9.4e-3)	1.75e-1(2.3e-3)	1.46e-1 (7.2e-3)	1.52e-1 (8.42e-4)	2.05e-1 (7.41e-3)	**1.34e-1 (2.4e-3)**
	8	3.92e-1(1.1e-2)	3.28e-1(1.8e-2)	2.90e-1 (2.1e-2)	2.25e-1 (3.3e-2)	3.02e-1 (8.57e-3)	**1.73e-1 (2.1e-2)**
MAF2	3	4.42e-2(1.6e-2)	3.24e-2(2.7e-3)	**2.98e-2(5.4e-3)**	3.15e-2(1.4e-3)	3.22e-2(1.0e-3)	3.27e-2(2.2e-3)
	5	1.22e-1(1.1e-2)	1.07e-1(1.3e-2)	9.03e-2(6.4e-2)	1.05e-1(1.4e-2)	1.22e-1(1.0e-3)	**8.6e-2(1.3e-3)**
	8	2.05e-01(1.3e-2)	1.63e-1(1.0e-2)	2.50e-1(1.4e-2)	1.69e-1(2.0e-3)	2.22e-1(2.4e-3)	**1.44e-1(1.6e-3)**
WFG1	3	1.29e-1(5.16e-2)	1.18e-1(4.4e-2)	1.17e-1(4.43e-2)	1.25e-1(3.13e-2)	4.95e-1(2.03e-2)	1.16e-1(7.43e-2)
	5	3.25e-1(2.2e-2)	**2.01e-1(1.5e-2)**	2.40e-1(6.6e-2)	3.75e-1(4.3e-2)	2.54e0(4.1e-2)	2.09e-1(1.4e-2)
	8	1.86e-1(9.2e-2)	1.77e-1(4.43e-2)	1.73e-1(2.03e-2)	1.71e-1(7.43e-2)	1.79e-1(3.6e-1)	1.72e-1(8.6e-1)
WFG2	3	8.02e-1(4.52e-2)	8.40e-1(3.9e-2)	8.66e-1(5.43e-3)	**6.75e-2(4.43e-3)**	6.95e-2(3.03e-3)	8.86e-1(6.43e-3)
	5	6.54e-1(2.2e-2)	6.47e-1(3.8e-2)	6.37e-1(8.5e-2)	6.85e-1(9.9e-2)	6.50e-1(6.6e-2)	**6.35e-1(3.9e-2)**
	8	1.42e0(9.1e-2)	1.26e0(7.9e-2)	2.04e0(8.9e-1)	**1.08e0(4.9e-2)**	1.50e0(4.6e-1)	1.15e0(3.6e-1)
WFG9	3	2.75e-01(8.35e-2)	2.350e-1(3.9e-2)	1.86e-1(3.43e-2)	3.25e-1(1.43e-2)	3.35e-1(2.53e-3)	**1.37e-1(3.43e-3)**
	5	9.75e-01(1.0e-2)	**9.37e-1(8.3e-2)**	9.65e-1(8.05e-2)	9.85e-1(1.0e-2)	9.88e-1(9.77e-2)	9.78e-1(9.05e-2)
	8	1.55e0(1.7e-1)	1.49e0(1.48e-1)	1.48e0(5.43e-1)	1.88e0(3.09e-1)	1.92e-1(3.03e-1)	**1.45e0(1.74e-1)**

case of three-objective, MARP-NSGA-III is slightly better in HV than A-NSGA-III, RVEA*, and A-NSGA-III. Relevant results on problem instances with three-objective problems can also be seen in Figures A.5 (a) to A.5 (c). The approximate Pareto-front of each algorithm shows that A-NSGA-III, RVEA*, and MARP-NSGA-III have a similar distribution of obtained points.



(a) ANSGA-III. (b) RVEA*. (c) MARP-NSGA-III.

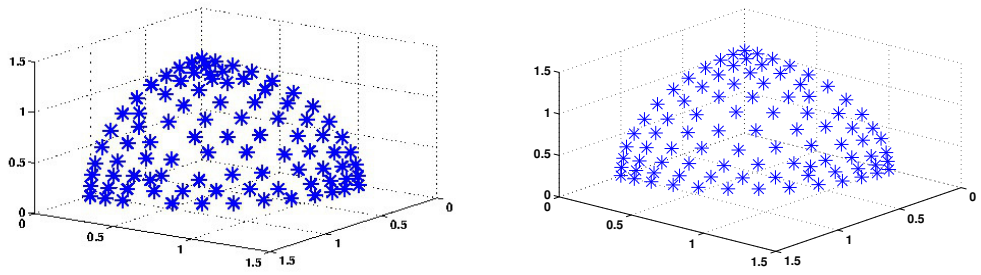
Figure A.5: Approximate Pareto-front for 3-objective IDTLZ1 problem

For the IDTLZ2 problem, Tables A.4 and A.5 show that IMMOEA achieved a significantly better result compared to all other competing algorithms in terms of HV and IGD with three-objective problems. However, IMMOEA performed poorly on problems with more than three objectives. In comparison, MARP-NSGA-III achieved the best performance among all the competing algorithms on five objectives in terms of HV. However, in terms of IGD, MARP-NSGA-III outperforms all the other competing algorithms on eight-objective problems.

Performance on DTLZ4, DTLZ5, and DTLZ7

DTLZ4 has a biased density (the density of solutions is different from one location to another) targeted points on the true Pareto-front. This problem is to verify whether many-objective algorithms can maintain a proper distribution of the solutions. Table A.4 confirms that MARP-NSGA-III achieves overall the best performance in terms of HV. However, Table A.5

reveals that MARP-NSGA-III is significantly better than the other competing algorithms on three-objective and five-objective. For three-objective, Figures A.6 (a) and A.6 (b) show that MARP-NSGA-III has widely distributed the solution as compared to A-NSGA-III. In terms of HV, the MARP-NSGA-III performed significantly better than the other compared algorithms on eight-objective problems. By contrast, A-NSGA-III has the best IGD value over eight-objective problems when compared to other competing algorithms.

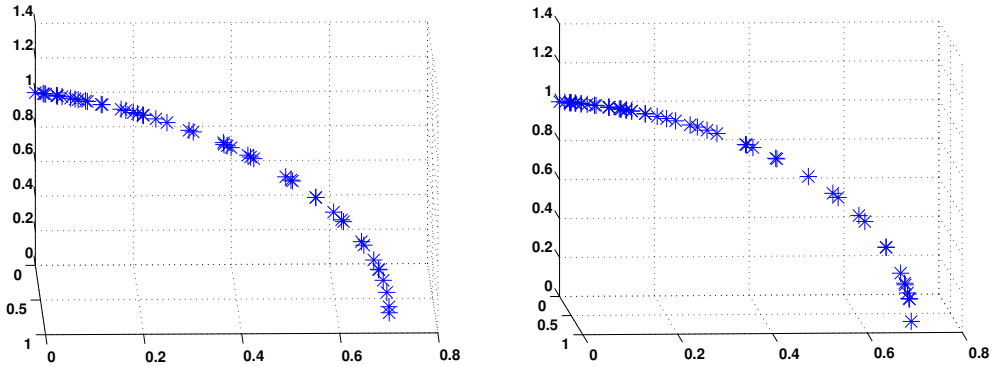


(a) A-NSGA-III. (b) MARP-NSGA-III.

Figure A.6: Approximate Pareto-front for 3-objective DTLZ4 problem

DTLZ5 has a degenerated Pareto-front, i.e., the Pareto-front is always a curve regardless of the objective space's dimensionality. For the three objective DTLZ5, Two-ARCh performed significantly better than other competing algorithms in terms of HV. The result can be observed in Table A.4. This result is also illustrated in Figures A.7 (a) and A.7 (b) where Two-ARCh has well-diversified optimal solutions. It can be observed in Tables A.4 and A.5 that no single algorithm can perform consistently better than the rest algorithms on DTLZ5.

DTLZ7 has a disconnected Pareto-front. Due to this feature, those algorithms that rely on uniformly distributed reference points performed poorly. Tables A.4 and A.5 reveal that the algorithms having adaptive reference points eventually have significantly better performance. In the



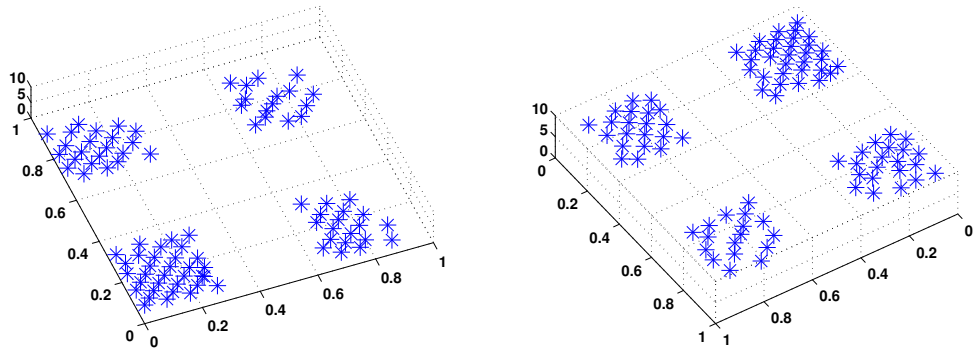
(a) Two-ARCh. (b) MARP-NSGA-III.

Figure A.7: Approximate Pareto-front for 3-objective DTLZ5 problem

three-objective case, our proposed algorithm (MARP-NSGA-III) is slightly better than Two-ARCh in terms of HV. However, MARP-NSGA-III outperforms RVEA* which can also be seen in Figures A.8 (a) and A.8 (b). These figures show that MARP-NSGA-III has well-diversified solutions in three out of four regions of the Pareto-front. RVEA* apparently failed to cover some regions on the Pareto-front. In five-objective and eight-objective cases, MARP-NSGA-III also performs the best which can be seen in Tables A.4 and A.5. Meanwhile, as evidenced in Figures A.9 (a) and A.9 (b), MARP-NSGA-III can evolve more diversified solutions than A-NSGA-III.

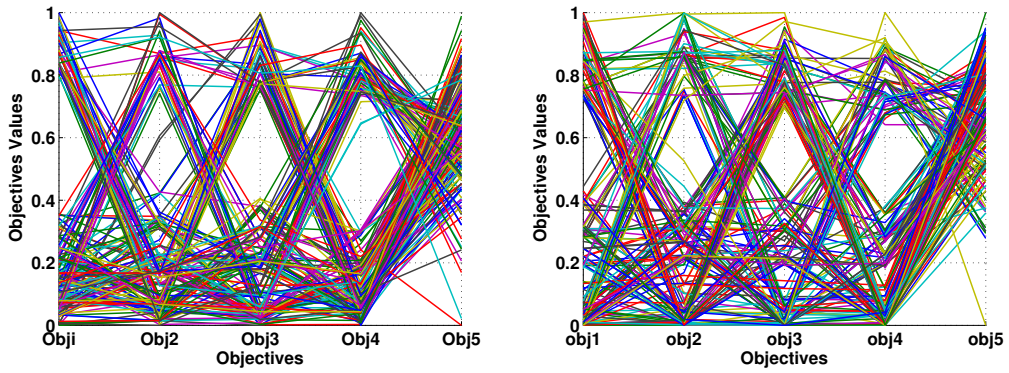
Performance on MAF1 and MAF2

MAF1 and MAF2 are taken from the CEC-2017 competition on MaOPs [32]. Moreover, they have irregular Pareto-front. MAF1 is a modified inverted DTLZ1 which has an inverted Pareto-front. This problem has the same structure as IDTLZ1. Therefore, we can see similar behavior of IDTLZ1 in the result. This can be seen in Tables A.4 and A.5. The experimental results reveal that the MARP-NSGA-III performed significantly better on problems with five and eight objectives than other competing



(a) RVEA*. (b) MARP-NSGA-III.

Figure A.8: Approximate Pareto-front for 3-objective DTLZ7 problem

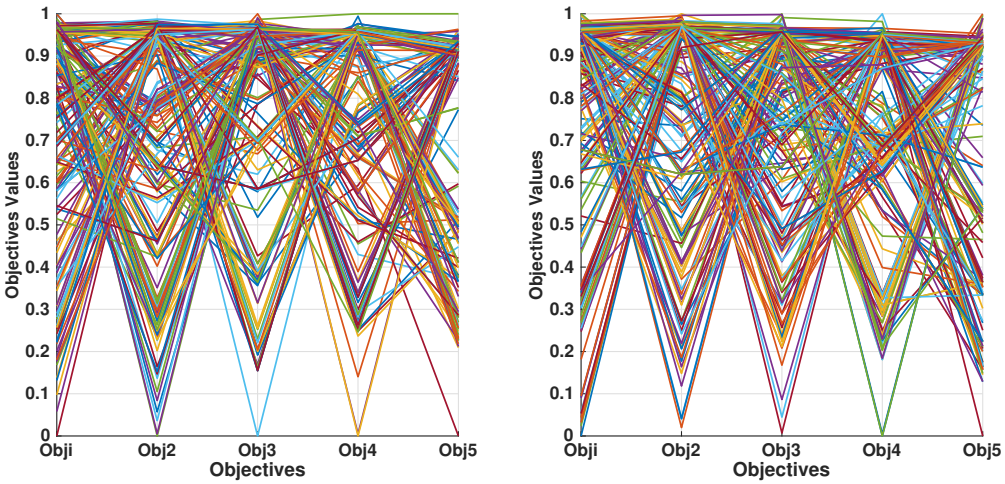


(a) A-NSGA-III. (b) MARP-NSGA-III.

Figure A.9: Parallel coordinate plot for the fitness values of the population for 5-objective DTLZ7 problem

algorithms. In Figures A.9 (a) and A.9 (b), the parallel plots show that optimal solutions obtained from MARP-NSGA-III are widely distributed.

MAF2 is a modified form of DTLZ2, with the higher irregularity of the Pareto-front. Tables A.4 and A.5 show that MARP-NSGA-III has achieved



(a) RVEA*. (b) MARP-NSGA-III.

Figure A.10: Parallel coordinate plot for the fitness values of the population for 5-objective MAF1 problem

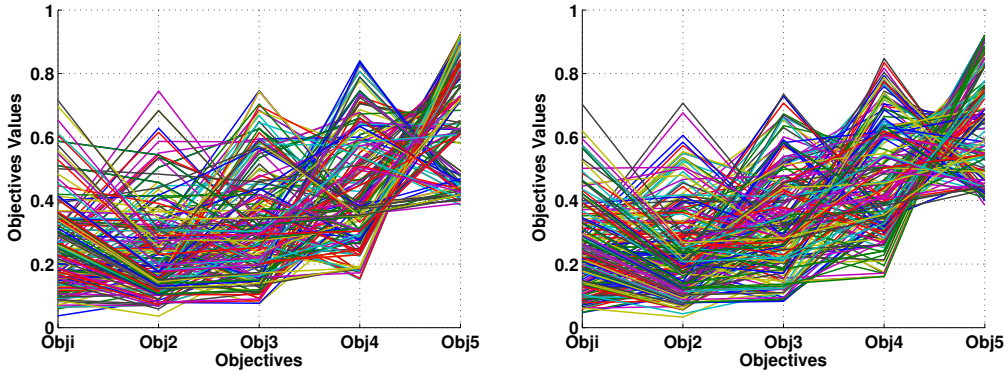
significantly better results on MAF2 as compared to the other algorithms on five-objective problems. We can see that in Figures A.11 (a) and A.11 (b), MARP-NSGA-III has better diversity than its competitor RVEA*.

In conclusion, MARP-NSGA-III outperforms the other algorithms on five-objective and eight-objective MAF problems.

Performance on WFG1,WFG2, and WFG9

Similar observations of MARP-NSGA-III can be made from the results on WFG in Table A.4 where MARP-NSGA-III outperforms the other algorithms on the eight-objective instances. RVEA* has also shown competitive performance on WFG problems.

WFG1 is a mixed structure problem involving many transformation functions in the problem definition. These transformation functions make it hard to produce well-diversified solutions [92]. We can observe from



(a) RVEA*

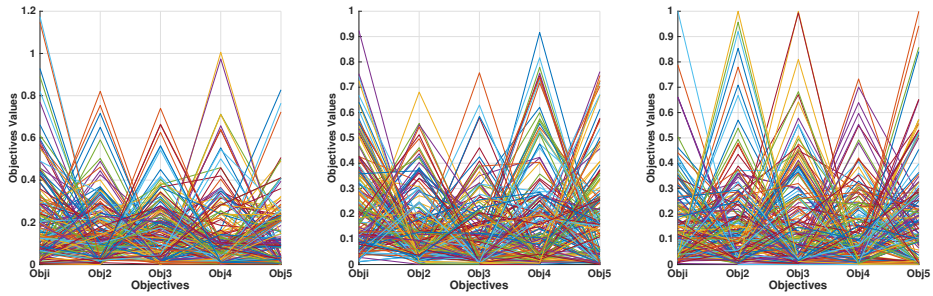
(b) MARP-NSGA-III.

Figure A.11: Parallel coordinate plot for the fitness values of the population for 5-objective MAF2 problem

Table A.4 and Table A.5 that RVEA* performs better than the other compared algorithms with three objectives problems, while A-NSGA-III outperforms the other algorithms on WFG1 with five-objective problems. In the case of eight objectives, our proposed algorithm MARP-NSGA-III achieved a higher HV value than other algorithms.

WFG2 is the only WFG problem with disconnected Pareto-front. It can be observed from Tables A.4 and A.5 that the MARP-NSGA-III outperforms the other algorithms with five-objective. This can be seen in Figures A.12 (a) to A.12 (c). While RVEA* has better performance than A-NSGA-III. It performed significantly worse performance than MARP-NSGA-III. This is because MARP-NSGA-III can produce well distributed and well-diversified solutions. In the meantime, MARP-NSGA-III achieves competitive performance as RVEA* on problem instances with eight objectives. MARP-NSGA-III shows promising performance on some eight-objective. Moreover, RVEA* shows generally competitive performance.

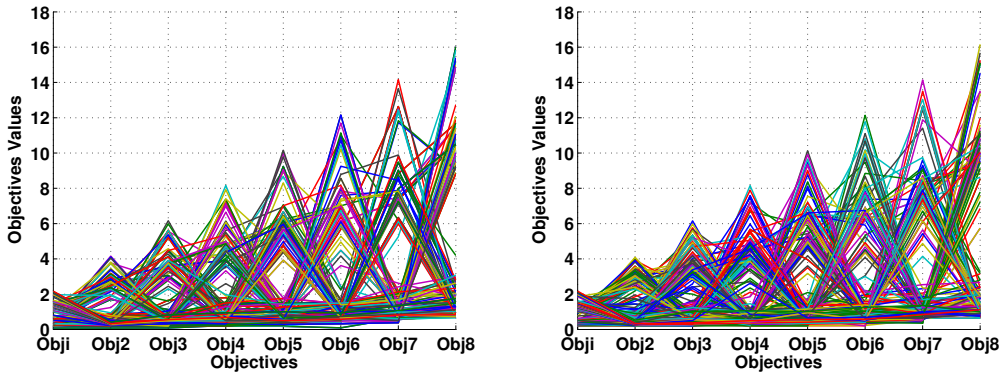
WFG9 has a scaled concave Pareto-front and its decision variables are non-separable. Tables A.4 and A.5 show that MARP-NSGA-III achieved



(a) A-NSGA-III. (b) RVEA*. (c) MARP-NSGA-III

Figure A.12: Parallel coordinate plot for the fitness values of the population for 5-objective WFG2 problem

leading performance on WFG9 with eight objectives. Figures A.13 (a) and A.13 (b) show that MARP-NSGA-III can produce more diversified solutions than RVEA*.



(a) RVEA*. (b) MARP-NSGA-III.

Figure A.13: Parallel coordinate plot for the fitness values of the population for eight-objective WFG9 problem

As we can observe in Tables A.4 and A.5, MARP-NSGA-III shows significantly better results on most of the five-objective and eight-objective

instances as compared to other algorithms. By contrast, RVEA* shows promising performance on some of the eight-objective instances. A-NSGA-III and Two-ARCh show effectiveness on three objective problems. Based on the experimental results, we can conclude that MARP-NSGA-III shows better performance on many benchmark problems. This algorithm performs particularly well on many irregular Pareto-front problems, such as inverted, disconnected, and scaling problems.

Bibliography

[1] AARTS, E., AND LENSTRA, J. K. *Local search in combinatorial optimization*. Wiley, Chichester, 1997.

[2] ABDUL-RAZAQ, T. S., POTTS, C. N., AND VAN WASSENHOVE, L. N. A survey of algorithms for the single machine total weighted tardiness scheduling problem . *Discrete Applied Mathematics 26*, 2–3 (1990), 235–253.

[3] ADAMS, J., BALAS, E., AND ZAWACK, D. The shifting bottleneck procedure for job shop scheduling. *Management science 34*, 3 (1988), 391–401.

[4] ADIBI, M. A., ZANDIEH, M., AND AMIRI, M. Multi-objective scheduling of dynamic job shop using variable neighborhood search. *Expert Syst. Appl. 37*, 1 (2010), 282–287.

[5] AITZAI, A., BENMEDJDOUB, B., AND BOUDHAR, M. Branch-and-bound and PSO algorithms for no-wait job shop scheduling. `http://dx.doi.org/10.1007/s10845-014-0906-7`, 2014.

[6] ALPAYDIN, E. *Introduction to Machine Learning*, 2 ed. MIT press, 2010.

[7] APPLEGATE, D., AND COOK, W. A Computational Study of the Job-Shop Scheduling Problem. *ORSA Journal on Computing 3*, 2 (1991), 149–156.

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

[9] ATTAR, S., MOHAMMADI, M., AND TAVAKKOLI-MOGHADDAM, R. Hybrid flexible flowshop scheduling problem with unrelated parallel machines and limited waiting times. *The International Journal of Advanced Manufacturing Technology 68*, 5-8 (2013), 1583–1599.

[10] AVANTHAY, C., HERTZ, A., AND ZUFFEREY, N. A variable neighborhood search for graph coloring. *European Journal of Operational Research 151*, 2 (2003), 379–388.

[11] BAKER, K. R., AND TRIETSCH, D. *Principles of sequencing and scheduling*. John Wiley & Sons, 2013.

[12] BENI, G., AND WANG, J. Swarm intelligence in cellular robotic systems. In *Robots and Biological Systems: Towards a New Bionics?* Springer, 1993, pp. 703–712.

[13] BŁAŻEWICZ, J., DOMSCHKE, W., AND PESCH, E. The job shop scheduling problem: Conventional and new solution techniques. *European Journal of Operational Research 93*, 1 (1996), 1–33.

[14] BLOT, A., JOURDAN, L., AND KESSACI, M.-É. Automatic design of multi-objective local search algorithms: case study on a bi-objective permutation flowshop scheduling problem. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2017), ACM, pp. 227–234.

[15] BOSMAN, P. A., AND THIERENS, D. The naive MIDEA: A baseline multi–objective EA, x-fetchedfrom = Google Scholar, year = 2005. In *International Conference on Evolutionary Multi-Criterion Optimization*, Springer, pp. 428–442.

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

[17] BRANKE, J., NGUYEN, S., PICKARDT, C., AND ZHANG, M. Automated Design of Production Scheduling Heuristics: A Review. *IEEE Trans. Evolutionary Computation 20*, 1 (2016), 110–124.

[18] BRINGMANN, K., AND FRIEDRICH, T. An Efficient Algorithm for Computing Hypervolume Contributions**. *Evol. Comput. 18*, 3 (Sept. 2010), 383–402.

[19] BRUCKER, P., JURISCH, B., AND SIEVERS, B. A branch and bound algorithm for the job-shop scheduling problem. *Discrete applied mathematics 49*, 1 (1994), 107–127.

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

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

[22] BYRNE, J., SVOROBEJ, S., GIANNOUTAKIS, K. M., TZOVARAS, D., BYRNE, P. J., ÖSTBERG, P.-O., GOURINOVITCH, A., AND LYNN, T. A review of cloud computing simulation platforms and related environments. In *International Conference on Cloud Computing and Services Science* (2017), vol. 2, SCITEPRESS, pp. 679–691.

[23] ÇALIŞ, B., AND BULKAN, S. A research survey: Review of AI solution strategies of job shop scheduling problem. *Journal of Intelligent Manufacturing 26*, 5 (2015), 961–973.

[24] CARLIER, J. The One-Machine Sequencing Problem. *European Journal of Operational Research 11*, 1 (1982), 42–47.

[25] CARLIER, J., AND PINSON, E. An Algorithm for Solving the Job-shop Problem. *Management Science 35*, 2 (1989), 164–176.

[26] CHEN, B., ZENG, W., LIN, Y., AND ZHANG, D. A new local search-based multiobjective optimization algorithm. *IEEE Transactions on Evolutionary Computation 19*, 1 (2015), 50–73.

[27] CHEN, S.-W., AND CHIANG, T.-C. Evolutionary many-objective optimization by MO-NSGA-II with enhanced mating selection. In *IEEE Congress on Evolutionary Computation* (2014), IEEE, pp. 1397–1404.

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

[29] CHENG, R., HE, C., JIN, Y., AND YAO, X. Model-based evolutionary algorithms: a short survey. *Complex & Intelligent Systems 4*, 4 (2018), 283–292.

[30] CHENG, R., JIN, Y., NARUKAWA, K., AND SENDHOFF, B. A multi-objective evolutionary algorithm using Gaussian process-based inverse modeling. *IEEE Transactions on Evolutionary Computation 19*, 6 (2015), 838–856.

[31] CHENG, R., JIN, Y., OLHOFER, M., AND SENDHOFF, B. A reference vector guided evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation 20*, 5 (2016), 773–791.

[32] CHENG, R., LI, M., TIAN, Y., ZHANG, X., YANG, S., JIN, Y., AND YAO, X. A benchmark test suite for evolutionary many-objective optimization. *Complex & Intelligent Systems 3*, 1 (2017), 67–81.

[33] CHOI, I.-C., AND CHOI, D.-S. A local search algorithm for jobshop scheduling problems with alternative operations and sequence-dependent setups. *Computers & Industrial Engineering 42*, 1 (2002), 43–58.

[34] COLORNI, A., DORIGO, M., MANIEZZO, V., AND TRUBIAN, M. Ant system for job-shop scheduling. *Belgian Journal of Operations Research, Statistics and Computer Science 34*, 1 (1994), 39–53.

[35] CORNELL, J. A. *Experiments with mixtures: designs, models, and the analysis of mixture data*, vol. 403. John Wiley & Sons, 2011.

[36] CORRIVEAU, G., GUILBAULT, R., TAHAN, A., AND SABOURIN, R. Bayesian network as an adaptive parameter setting approach for genetic algorithms. *Complex & Intelligent Systems 2*, 1 (2016), 1–22.

[37] COWLING, P., KENDALL, G., AND SOUBEIGA, E. A Hyperheuristic Approach to Scheduling a Sales Summit. In *Practice and Theory of Automated Timetabling III*, vol. 2079 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2001, pp. 176–190.

[38] CROES, A. A method for solving traveling salesman problems. *Operations Research 5* (1958), 791—812.

[39] DAS, I., AND DENNIS, J. E. Normal-boundary intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems. *SIAM Journal on Optimization 8*, 3 (1998), 631–657.

[40] DAVIS, L. *Evolutionary algorithms*. The IMA volumes in mathematics and its applications. Springer, 1999.

[41] DE BONET, J. S., ISBELL JR, C. L., AND VIOLA, P. A. MIMIC: Finding optima by estimating probability densities. In *Advances in neural information processing systems* (1997), pp. 424–430.

[42] DEB, K. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, 2001.

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

[44] DEB, K., PRATAP, A., AGARWAL, S., AND MEYARIVAN, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation 6* (2002), 182–197.

[45] DEB, K., THIELE, L., AND ZITZLER, E. Scalable multi-objectove optimization test problems. In *IEEE Congress on Evolutionary Computation* (2002), IEEE, pp. 825–830.

[46] DEMIRKOL, E., MEHTA, S., AND UZSOY, R. Benchmarks for shop scheduling problems. *European Journal of Operational Research 109*, 1 (1998), 137–141.

[47] DICK, G. Sensitivity-like analysis for feature selection in genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2017), pp. 401–408.

[48] DRAGOMIR, S. S., CERONE, P., AND SOFO, A. Some remarks on the midpoint rule in numerical integration. *RGMIA research report collection 1*, 2 (1998).

[49] DUBOIS-LACOSTE, J., LÓPEZ-IBÁÑEZ, M., AND STÜTZLE, T. A hybrid TP+PLS algorithm for bi-objective flow-shop scheduling problems. *Computers & OR 38*, 8 (2011), 1219–1236.

[50] DUBOIS-LACOSTE, J., LÓPEZ-IBÁÑEZ, M., AND STÜTZLE, T. Anytime pareto local search. *European Journal of Operational Research 243*, 2 (2015), 369–385.

[51] FOWLER, L., PFUND, M., YU, L., FOWLER, J. W., AND CARLYLE, W. M. Development Of A Robust Scheduling Rule For A Printed Wiring Board Drilling Operation With Multiple Scheduling Objectives And Fixed Order Release/Pickup Times. In *IIE Annual Conference. Proceedings* (2002), Citeseer, p. 1.

[52] GALVÁN-LÓPEZ, E., MCDERMOTT, J., O'NEILL, M., AND BRABAZON, A. Defining locality as a problem difficulty measure in genetic programming. *Genetic Programming and Evolvable Machines 12*, 4 (2011), 365–401.

[53] GAREY, M. R., JOHNSON, D. S., AND SETHI, R. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research 1*, 2 (1976), 117–129.

[54] GENG, S., WU, D., WANG, P., AND CAI, X. Many-objective cloud task scheduling. *IEEE Access 8* (2020), 79079–79088.

[55] GIFFLER, B., AND THOMPSON, G. L. Algorithms for solving production-scheduling problems. *Operations research 8*, 4 (1960), 487–503.

[56] GONG, X., DE PESSEMIER, T., MARTENS, L., AND JOSEPH, W. Energy-and labor-aware flexible job shop scheduling under dynamic electricity pricing: A many-objective optimization investigation. *Journal of Cleaner Production 209* (2019), 1078–1094.

[57] GONG, X., DE PESSEMIER, T., MARTENS, L., AND JOSEPH, W. Energy-and labor-aware flexible job shop scheduling under dynamic electricity pricing: A many-objective optimization investigation. *Journal of cleaner production 209* (2019), 1078–1094.

[58] GONZÁLEZ RODRÍGUEZ, I., RODRÍGUEZ VELA, M. D. C., PUENTE PEINADOR, J., AND HERNÁNDEZ ARAUZO, A. Improved local search for job shop scheduling with uncertain durations. In *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling* (2009), Association for the Advancement of Artificial Intelligence (AAAI).

[59] GRIFFIN, J. L., SCHLOSSER, S. W., GANGER, G. R., AND NAGLE, D. *Operation Management*. Cengage Learning, 2015.

[60] GROMICHO, J. A., VAN HOORN, J. J., SALDANHA-DA GAMA, F., AND TIMMER, G. T. Solving the job-shop scheduling problem optimally by dynamic programming . *Computers & Operations Research 39*, 12 (2012), 2968–2977.

[61] GUPTA, A. K., AND SIVAKUMAR, A. I. Job shop scheduling techniques in semiconductor manufacturing. *The International Journal of Advanced Manufacturing Technology 27*, 11-12 (2006), 1163–1169.

[62] HART, E., ROSS, P., AND CORNE, D. Evolutionary Scheduling: A Review. *Genetic Programming and Evolvable Machines 6*, 2 (2005), 191–220.

[63] HEGER, J., HILDEBRANDT, T., AND SCHOLZ-REITER, B. Dispatching rule selection with gaussian processes. *Central European Journal of Operations Research 23*, 1 (2015), 235–249.

[64] HELD, M., AND KARP, R. M. A Dynamic Programming Approach to Sequencing Problems. In *Proceedings of the 16th ACM National Meeting (ACM 1961)* (1961), pp. 71.201–71.204.

[65] HERRERO, J. G., BERLANGA, A., AND LOPEZ, J. M. M. Effective evolutionary algorithms for many-specifications attainment: Application to air traffic control tracking filters. *Evolutionary Computation, IEEE Transactions on 13*, 1 (2009), 151–168.

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

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

[68] HO, N., AND TAY, J. Evolving dispatching rules for solving the flexible job-shop problem. In *IEEE Congress on Evolutionary Computation* (2005), vol. 3, IEEE, pp. 2848–2855.

[69] HOLLAND, J. *Adaption in Natural and Artificial Systems*. MIT Press, Cambridge, MA, 1992. 1st edition: 1975, The University of Michigan Press, Ann Arbor.

[70] HOLTHAUS, O., AND RAJENDRAN, C. Efficient jobshop dispatching rules: Further developments. *Production Planning & Control 11*, 2 (2000), 171–178.

[71] HORST, R., AND ROMEIJN, H. E. *Handbook of Global Optimization*, vol. 2. Springer Science & Business Media, 2002.

[72] HUGHES, E. J. Multiple single objective Pareto sampling. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on* (2003), vol. 4, IEEE, pp. 2678–2684.

[73] HUNT, R., JOHNSTON, M., AND ZHANG, M. Evolving "less-myopic" scheduling rules for dynamic job shop scheduling with genetic programming. In *GECCO* (2014), D. V. Arnold, Ed., ACM, pp. 927–934.

[74] HUNT, R., JOHNSTON, M., AND ZHANG, M. Evolving Dispatching Rules with Greater Understandability for Dynamic Job Shop Scheduling. Tech. rep., Victoria University, June 2015.

[75] HUNT, R., JOHNSTON, M., AND ZHANG, M. Using Local Search to Evaluate Dispatching Rules in Dynamic Job Shop Scheduling. In *EvoCOP* (2015), G. Ochoa and F. Chicano, Eds., vol. 9026 of *Lecture Notes in Computer Science*, Springer, pp. 222–233.

[76] IBRAHIM, A., RAHNAMAYAN, S., MARTIN, M. V., AND DEB, K. EliteNSGA-III: An improved evolutionary many-objective optimization algorithm. In *CEC* (2016), IEEE, pp. 973–982.

[77] IKEDA, K.-I., KITA, H., AND KOBAYASHI, S. Failure of Pareto-based MOEAs: does non-dominated really mean near to optimal? In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on* (2001), vol. 2, IEEE, pp. 957–962.

[78] ISHIBUCHI, H., IMADA, R., SETOGUCHI, Y., AND NOJIMA, Y. Reference point specification in hypervolume calculation for fair comparison and efficient search. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2017), pp. 585–592.

[79] ISHIBUCHI, H., AND MURATA, T. A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 28*, 3 (1998), 392–403.

[80] ISHIBUCHI, H., SETOGUCHI, Y., MASUDA, H., AND NOJIMA, Y. Performance of decomposition-based many-objective algorithms strongly depends on Pareto front shapes. *IEEE Transactions on Evolutionary Computation 21*, 2 (2016), 169–190.

[81] ISHIBUCHI, H., AND YOSHIDA, T. Hybrid Evolutionary Multi-Objective Optimization Algorithms. In *HIS* (2002), A. Abraham, J. R. del Solar, and M. Köppen, Eds., vol. 87 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, pp. 163–172.

[82] ISHIBUCHI, H., YOSHIDA, T., AND MURATA, T. Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Trans. Evolutionary Computation 7*, 2 (2003), 204–223.

[83] JACKSON, J. An extension of Johnson's result on job-lot scheduling. *Naval Research Logistics Quarterly 3*, 3 (1956), 201–204.

[84] JAIMES, A. L., AND COELLO, C. A. C. Many-Objective Problems: Challenges and Methods. In *Handbook of Computational Intelligence*, J. Kacprzyk and W. Pedrycz, Eds. Springer, 2015, pp. 1033–1046.

[85] JAIN, H., AND DEB, K. An Improved Adaptive Approach for Elitist Nondominated Sorting Genetic Algorithm for Many-Objective Optimization. In *EMO* (2013), R. C. Purshouse, P. J. Fleming, C. M. Fonseca, S. Greco, and J. Shaw, Eds., vol. 7811 of *Lecture Notes in Computer Science*, Springer, pp. 307–321.

[86] JAIN, H., AND DEB, K. An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point Based Nondominated Sorting Approach, Part II: Handling Constraints and Extending to an Adaptive Approach. *IEEE Transactions. Evolutionary Computation 18*, 4 (2014), 602–622.

[87] JAKOBOVIĆ, D., AND MARASOVIĆ, K. Evolving priority scheduling heuristics with genetic programming. *Applied Soft Computing 12*, 9 (2012), 2781–2789.

[88] JASZKIEWICZ, A. On the performance of multiple-objective genetic local search on the 0/1 knapsack problem - a comparative experiment. *IEEE Transactions on Evolutionary Computation 6*, 4 (Aug. 2002), 402–412.

[89] JASZKIEWICZ, A. Many-Objective Pareto Local Search. *European Journal of Operational Research 271*, 3 (2018), 1001–1013.

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

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

[92] JIANG, S., AND YANG, S. A strength Pareto evolutionary algorithm based on reference direction for multiobjective and many-objective optimization. *IEEE Transactions on Evolutionary Computation 21*, 3 (2017), 329–346.

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

[94] JONES, A., RABELO, L. C., AND SHARAWI, A. T. Survey of job shop scheduling techniques. *Wiley encyclopedia of electrical and electronics engineering* (2001).

[95] JONG, K. D. *Evolutionary Computation: A Unified Approach*. The MIT Press, 2006.

[96] KACEM, I., HAMMADI, S., AND BORNE, P. Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic. *Mathematics and computers in simulation 60*, 3 (2002), 245–276.

[97] KARUNAKARAN, D., CHEN, G., AND ZHANG, M. Parallel multi-objective job shop scheduling using genetic programming. In *Australasian Conference on Artificial Life and Computational Intelligence* (2016), Springer, pp. 234–245.

[98] KARUNAKARAN, D., MEI, Y., CHEN, G., AND ZHANG, M. Sampling heuristics for multi-objective dynamic job shop scheduling using island based parallel genetic programming. In *International Conference on Parallel Problem Solving from Nature* (2018), Springer, pp. 347–359.

[99] KASHAN, A. H., KESHMIRY, M., DAHOOIE, J. H., AND ABBASI-POOYA, A. A simple yet effective grouping evolutionary strategy (ges) algorithm for scheduling parallel machines. *Neural Computing and Applications 30*, 6 (2018), 1925–1938.

[100] KENNEDY, J., AND EBERHART, R. Particle swarm optimization. In *Proceedings of IEEE international conference on neural networks* (1995), vol. 4, Perth, Australia, pp. 1942–1948.

[101] KENNEDY, J., AND EBERHART, R. C. *Swarm Intelligence*. Morgan Kaufmann, April 2001.

[102] KNOWLES, J., AND CORNE, D. M-PAES: a memetic algorithm for multiobjective optimization. In *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)* (July 2000), vol. 1, IEEE, pp. 325–332 vol.1.

[103] KOLAHAN, F., AND KAYVANFAR, V. A heuristic algorithm approach for scheduling of multi-criteria unrelated parallel machines. *World, Academy of Science, Engineering and Technology 59* (2009), 102.

[104] KOZA, J. R. A Genetic Approach to the Truck Backer Upper Problem and the Inter-Twined Spiral Problem. In *Proceedings of IJCNN International Joint Conference on Neural Networks* (1992), vol. IV, IEEE Press, pp. 310–318.

[105] KOZA, J. R. *Genetic Programming – On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[106] KREIPL, S. A large step random walk for minimizing total weighted tardiness in a job shop. *Journal of Scheduling 3*, 3 (2000), 125–138.

[107] LAND, A. H., AND DOIG, A. G. An automatic method for solving discrete programming problems. *Econometrica* (1960), 497–520.

[108] LARA, A., SANCHEZ, G., COELLO, C. A. C., AND SCHÜTZE, O. HCS: A New Local Search Strategy for Memetic Multiobjective Evolutionary Algorithms. *IEEE Trans. Evolutionary Computation 14*, 1 (2010), 112–132.

[109] LARRAÑAGA, P., AND LOZANO, J. A. *Estimation of distribution algorithms: A new tool for evolutionary computation*, vol. 2. Springer Science & Business Media, 2001.

[110] LAUMANNS, M., AND OCENASEK, J. Bayesian optimization algorithms for multi-objective optimization. In *International Conference on Parallel Problem Solving from Nature* (2002), Springer, pp. 298–307.

[111] LAWLER, E. L., LENSTRA, J. K., RINNOOY KAN, A. H. G., AND SHMOYS, D. B. Sequencing and Scheduling: Algorithms and Complexity . In *Logistics of Production and Inventory*, vol. 4 of *Handbooks in Operations Research and Management Science* . Elsevier, 1993, pp. 445–522.

[112] LAWLER, E. L., AND MOORE, J. M. A Functional Equation and Its Application to Resource Allocation and Sequencing Problems. *Management Science 16*, 1 (1969), 77–84.

[113] LAWRENCE, S. Resouce constrained project scheduling: An experimental investigation of heuristic scheduling techniques (Supplement). *Graduate School of Industrial Administration, Carnegie-Mellon University* (1984).

[114] LEE, Y. H., BHASKARAN, K., AND PINEDO, M. A heuristic to minimize the total weighted tardiness with sequence-dependent setups. *IIE transactions 29*, 1 (1997), 45–52.

[115] LEUNG, J. Y.-T., Ed. *Handbook of Scheduling - Algorithms, Models, and Performance Analysis.* Chapman and Hall/CRC, 2004.

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

[117] LI, K., KWONG, S., CAO, J., LI, M., ZHENG, J., AND SHEN, R. Achieving balance between proximity and diversity in multi-objective evolutionary algorithm. *Inf. Sci. 182*, 1 (2012), 220–242.

[118] LI, M. Pareto or non-Pareto: Bi-criterion evolution in multiobjective optimization. *IEEE Transactions on Evolutionary Computation 20*, 5 (2015), 645–665.

[119] LI, M., YANG, S., AND LIU, X. Shift-Based Density Estimation for Pareto-Based Algorithms in Many-Objective Optimization. *IEEE Trans. Evolutionary Computation 18*, 3 (2014), 348–365.

[120] LI, X., MABU, S., AND HIRASAWA, K. A novel graph-based estimation of the distribution algorithm and its extension using reinforcement learning. *Evolutionary Computation, IEEE Transactions on 18*, 1 (2014), 98–113.

[121] LIEFOOGHE, A., HUMEAU, J., MESMOUDI, S., JOURDAN, L., AND TALBI, E.-G. On dominance-based multiobjective local search: design, implementation and experimental analysis on scheduling and traveling salesman problems. *Journal of Heuristics 18*, 2 (2012), 317–352.

[122] LIN, S.-W., AND YING, K.-C. Minimizing makespan and total flowtime in permutation flowshops by a bi-objective multi-start

simulated-annealing algorithm. *Computers & OR 40*, 6 (2013), 1625–1647.

[123] LIU, Y., GONG, D., SUN, X., AND ZHANG, Y. A reference points-based evolutionary algorithm for many-objective optimization. In *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation* (2014), ACM, pp. 1053–1056.

[124] LOURENÇO, H. R., MARTIN, O. C., AND STÜTZLE, T. Iterated local search: Framework and applications. In *Handbook of metaheuristics*. Springer, 2019, pp. 129–168.

[125] LUKE, S., ET AL. A Java-based Evolutionary Computation Research System . `https://cs.gmu.edu/~eclab/projects/ecj/`.

[126] LUO, Q., DENG, Q., GONG, G., ZHANG, L., HAN, W., AND LI, K. An efficient memetic algorithm for distributed flexible job shop scheduling problem with transfers. *Expert Systems with Applications 160* (2020), 113721.

[127] LUST, T., AND TEGHEM, J. The multiobjective multidimensional knapsack problem: a survey and a new approach, July 2010.

[128] MASOOD, A., MEI, Y., CHEN, G., AND ZHANG, M. A PSO-Based Reference Point Adaption Method for Genetic Programming Hyper-Heuristic in Many-Objective Job Shop Scheduling. In *ACALCI* (2017), M. Wagner, X. Li, and T. Hendtlass, Eds., vol. 10142 of *Lecture Notes in Computer Science*, pp. 326–338.

[129] MATTFELD, D. C., AND BIERWIRTH, C. An efficient genetic algorithm for job shop scheduling with tardiness objectives. *European Journal of Operational Research 155*, 3 (2004), 616–630.

[130] MEI, Y., NGUYEN, S., XUE, B., AND ZHANG, M. An efficient feature selection algorithm for evolving job shop scheduling rules with

genetic programming. *IEEE Transactions on Emerging Topics in Computational Intelligence 1*, 5 (2017), 339–353.

[131] MEI, Y., ZHANG, M., AND NYUGEN, S. Feature Selection in Evolving Job Shop Dispatching Rules with Genetic Programming, in GECCO, ACM, 2016.

[132] MENCHACA-MENDEZ, A., MONTERO, E., AND MARTÍNEZ, S. Z. An Improved S-Metric Selection Evolutionary Multi-Objective Algorithm With Adaptive Resource Allocation. *IEEE Access 6* (2018), 63382–63401.

[133] MICHALEWICZ, Z., AND FOGEL, D. B. *How to Solve It: Modern Heuristics*, 2 ed. Springer Science & Business Media, 2013.

[134] MITCHELL, T. M. *Machine learning*. McGraw-Hill, New York, NY [u.a., 2010.

[135] MIYASHITA, K. Job-Shop Scheduling with GP. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)* (Las Vegas, Nevada, USA, July 2000), D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H.-G. Beyer, Eds., Morgan Kaufmann, pp. 505–512.

[136] MOSLEHI, G., AND MAHNAM, M. A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. *International Journal of Production Economics 129*, 1 (2011), 14–22.

[137] MURATA, T., ISHIBUCHI, H., AND TANAKA, H. Multi-objective genetic algorithm and its applications to flowshop scheduling. *Computers & Industrial Engineering 30*, 4 (1996), 957–968.

[138] MUSSELMAN, K., AND TALAVAGE, J. A tradeoff cut approach to multiple objective optimization. *Operations Research 28*, 6 (1980), 1424–1435.

[139] MUTH, J. F., AND THOMPSON, G. L. *Industrial Scheduling*. Prentice-Hall, 1963.

[140] NARUKAWA, K., AND RODEMANN, T. Examining the performance of evolutionary many-objective optimization algorithms on a real-world application. In *2012 Sixth International Conference on Genetic and Evolutionary Computing* (2012), IEEE, pp. 316–319.

[141] NGUYEN, S. *Automatic Design of Dispatching Rules for Job Shop Scheduling with Genetic Programming*. PhD thesis, 2013.

[142] NGUYEN, S., MEI, Y., MA, H., CHEN, A., AND ZHANG, M. Evolutionary scheduling and combinatorial optimisation: Applications, challenges, and future directions. In *2016 IEEE Congress on Evolutionary Computation (CEC)* (July 2016), IEEE, pp. 3053–3060.

[143] NGUYEN, S., ZHANG, M., AND JOHNSTON, M. A genetic programming based hyper-heuristic approach for combinatorial optimisation. In *GECCO* (2011), N. Krasnogor and P. L. Lanzi, Eds., ACM, pp. 1299–1306.

[144] NGUYEN, S., ZHANG, M., AND JOHNSTON, M. A sequential genetic programming method to learn forward construction heuristics for order acceptance and scheduling. In *2014 IEEE congress on evolutionary computation (CEC)* (2014), IEEE, pp. 1824–1831.

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

[146] NGUYEN, S., ZHANG, M., JOHNSTON, M., AND TAN, K. C. Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming. *IEEE Transactions on Evolutionary Computation 18*, 2 (2013), 193–208.

[147] NGUYEN, S., ZHANG, M., JOHNSTON, M., AND TAN, K. C. Dynamic multi-objective job shop scheduling: A genetic programming approach. In *Automated Scheduling and Planning*. Springer, 2013, pp. 251–282.

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

[149] NGUYEN, S., ZHANG, M., JOHNSTON, M., AND TAN, K. C. Automatic programming via iterated local search for dynamic job shop scheduling. *IEEE Transactions on Cybernetics 45*, 1 (2015), 1–14.

[150] OK, S., MIYASHITA, K., AND NISHIHARA, S. Improving performance of GP by adaptive terminal selection. In *PRICAI 2000 Topics in Artificial Intelligence*. Springer, 2000, pp. 435–445.

[151] OKABE, T., JIN, Y., SENDOFF, B., AND OLHOFER, M. Voronoi-based estimation of distribution algorithm for multi-objective optimization. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753)* (2004), vol. 2, IEEE, pp. 1594–1601.

[152] OSMAN, I. H., AND KELLY, J. P. *Meta-heuristics: An Overview*. Springer, 1996.

[153] OWEN, A., AND HARVEY, I. Adapting Particle Swarm Optimisation for Fitness Landscapes with Neutralit. IEEE Swarm Intelligence Symposium, IEEE.

[154] PANWALKAR, S. S., AND ISKANDER, W. A survey of scheduling rules. *Operations research 25*, 1 (1977), 45–61.

[155] PAQUETE, L., CHIARANDINI, M., AND STÜTZLE, T. Pareto local optimum sets in the biobjective traveling salesman problem: An

experimental study. In *Metaheuristics for multiobjective optimisation*. Springer, 2004, pp. 177–199.

[156] PAQUETE, L., AND STÜTZLE, T. A study of stochastic local search algorithms for the biobjective QAP with correlated flow matrices. *European Journal of Operational Research 169*, 3 (2006), 943–959.

[157] PARK, J., NGUYEN, S., ZHANG, M., AND JOHNSTON, M. Genetic programming for order acceptance and scheduling. In *2013 IEEE Congress on Evolutionary Computation* (2013), IEEE, pp. 1005–1012.

[158] PARK, J., NGUYEN, S., ZHANG, M., AND JOHNSTON, M. A Single Population Genetic Programming based Ensemble Learning Approach to Job Shop Scheduling. In *Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference* (2015), ACM, pp. 1451–1452.

[159] PARK, J., NGUYEN, S., ZHANG, M., AND JOHNSTON, M. Evolving Ensembles of Dispatching Rules Using Genetic Programming for Job Shop Scheduling. In *Genetic Programming*. Springer, 2015, pp. 92–104.

[160] PELIKAN, M., GOLDBERG, D. E., AND CANTU-PAZ, E. Linkage problem, distribution estimation, and Bayesian networks. *Evolutionary computation 8*, 3 (2000), 311–340.

[161] PELIKAN, M., GOLDBERG, D. E., AND LOBO, F. G. A survey of optimization by building and using probabilistic models. *Computational optimization and applications 21*, 1 (2002), 5–20.

[162] PENG, B., LÜ, Z., AND CHENG, T. A tabu search/path relinking algorithm to solve the job shop scheduling problem. *Computers & Operations Research 53* (2015), 154–164.

[163] PFUND, M., FOWLER, J. W., AND GUPTA, J. N. A survey of algorithms for single and multi-objective unrelated parallel-machine deterministic scheduling problems. *Journal of the Chinese Institute of Industrial Engineers 21*, 3 (2004), 230–241.

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

[165] PINEDO, M. L. *Scheduling: theory, algorithms, and systems*. Springer Science & Business Media, 2012.

[166] POTTS, C. N., AND STRUSEVICH, V. A. Fifty years of scheduling: a survey of milestones. *JORS 60*, S1 (2009).

[167] PRADITWONG, K., AND YAO, X. A new multi-objective evolutionary optimisation algorithm: the two-archive algorithm. In *Computational intelligence and security, 2006 international conference on* (2006), vol. 1, IEEE, pp. 286–291.

[168] RAGHU, T., AND RAJENDRAN, C. An efficient dynamic dispatching rule for scheduling in a job shop . *International Journal of Production Economics 32*, 3 (1993), 301–313.

[169] RAJENDRAN, C., AND HOLTHAUS, O. A comparative study of dispatching rules in dynamic flowshops and jobshops. *European journal of operational research 116*, 1 (1999), 156–170.

[170] RASMUSSEN, C. E., AND WILLIAMS, C. K. I. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[171] ROTHLAUF, F., AND OETZEL, M. On the locality of grammatical evolution. In *European Conference on Genetic Programming* (2006), Springer, pp. 320–330.

[172] RUSSELL, S. J., NORVIG, P., AND RUSSELL, S. J. *Artificial Intelligence: A Modern Approach (Prentice Hall Series in Artificial Intelligence)*, 0002 ed. Prentice Hall, 2003.

[173] SABAR, N. R., TURKY, A., AND SONG, A. A genetic programming based iterated local search for software project scheduling. In *GECCO* (2018), H. E. Aguirre and K. Takadama, Eds., ACM, pp. 1364–1370.

[174] SATO, H., AGUIRRE, H. E., AND TANAKA, K. Genetic Diversity and Effective Crossover in Evolutionary Many-objective Optimization. In *LION* (2011), C. A. C. Coello, Ed., vol. 6683 of *Lecture Notes in Computer Science*, Springer, pp. 91–105.

[175] SEADA, H. A., ABOUHAWWASH, M., AND DEB, K. Towards a Better Diversity of Evolutionary Multi-Criterion Optimization Algorithms Using Local Searches. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion* (New York, NY, USA, 2016), GECCO '16 Companion, ACM, pp. 77–78.

[176] SELS, V., GHEYSEN, N., AND VANHOUCKE, M. A comparison of priority rules for the job shop scheduling problem under different flow time-and tardiness-related objective functions. *International Journal of Production Research 50*, 15 (2012), 4255–4270.

[177] SHA, D., AND HSU, C.-Y. A hybrid particle swarm optimization for job shop scheduling problem . *Computers & Industrial Engineering 51*, 4 (2006), 791–808.

[178] SHAO, L., LIU, L., AND LI, X. Feature learning for image classification via multiobjective genetic programming. *Neural Networks and Learning Systems, IEEE Transactions on 25*, 7 (2014), 1359–1371.

[179] SPRECHER, A., KOLISCH, R., AND DREXL, A. Semi-active, active, and non-delay schedules for the resource-constrained project

scheduling problem . *European Journal of Operational Research 80*, 1 (1995), 94–102.

[180] SÜLFLOW, A., DRECHSLER, N., AND DRECHSLER, R. Robust multi-objective optimization in high dimensional spaces. In *International conference on evolutionary multi-criterion optimization* (2007), Springer, pp. 715–726.

[181] SURESH, V., AND CHAUDHURI, D. Dynamic scheduling – a survey of research. *International Journal of Production Economics 32*, 1 (1993), 53–63.

[182] TAILLARD, E. Benchmarks for basic scheduling problems. *european journal of operational research 64*, 2 (1993), 278–285.

[183] TAILLARD, E. D. Parallel taboo search techniques for the job shop scheduling problem. *ORSA journal on Computing 6*, 2 (1994), 108–117.

[184] TAVAKKOLI-MOGHADDAM, R., RAHIMI-VAHED, A., AND MIRZAEI, A. H. A hybrid multi-objective immune algorithm for a flow shop scheduling problem with bi-objectives: Weighted mean completion time and weighted mean tardiness. *Information Sciences 177*, 22 (2007), 5072–5090.

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

[186] THIERENS, D., AND BOSMAN, P. A. Multi-objective mixture-based iterated density estimation evolutionary algorithms. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation* (2001), Morgan Kaufmann Publishers Inc., pp. 663–670.

[187] TIAN, Y., CHENG, R., ZHANG, X., AND JIN, Y. PlatEMO: A MATLAB Platform for Evolutionary Multi-Objective Optimization [Educational Forum], Nov. 2017.

[188] TSAI, C.-W., AND RODRIGUES, J. J. P. C. Metaheuristic Scheduling for Cloud: A Survey. *IEEE Systems Journal 8*, 1 (2014), 279–291.

[189] VAESSENS, R. J. M., AARTS, E. H., AND LENSTRA, J. K. Job shop scheduling by local search. *INFORMS Journal on Computing 8*, 3 (1996), 302–317.

[190] VAN LAARHOVEN, P. J., AARTS, E. H., AND LENSTRA, J. K. Job shop scheduling by simulated annealing. *Operations research 40*, 1 (1992), 113–125.

[191] VEPSALAINEN, A. P. J., AND MORTON, T. E. Priority Rules for Job Shops with Weighted Tardiness Costs. *Management Science 33*, 8 (1987), 1035–1047.

[192] WANG, C., JI, Z., AND WANG, Y. Many-objective flexible job shop scheduling using nsga-iii combined with multi-attribute decision making. *Modern Physics Letters B 32*, 34n36 (2018), 1840110.

[193] WANG, H., JIAO, L., AND YAO, X. Two_Arch2: An Improved Two-Archive Algorithm for Many-Objective Optimization. *IEEE Transactions on Evolutionary Computation 19*, 4 (Aug. 2015), 524–541.

[194] WIDMER, G., AND KUBAT, M. Learning in the Presence of Concept Drift and Hidden Contexts. *Machine Learning 23*, 1 (1996), 69–101.

[195] WILCOXON, F. *Individual comparisons by ranking methods*. Springer, 1992.

[196] WONG, T. C., AND NGAN, S. C. A comparison of hybrid genetic algorithm and hybrid particle swarm optimization to minimize makespan for assembly job shop. *Applied Soft Computing 13*, 3 (2013), 1391–1399.

[197] XIA, W., AND WU, Z. An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering 48*, 2 (2005), 409–425.

[198] XU, W.-J., HE, L.-J., AND ZHU, G.-Y. Many-objective flow shop scheduling optimisation with genetic algorithm based on fuzzy sets. *International Journal of Production Research* (2019), 1–25.

[199] YANG, S., LI, M., LIU, X., AND ZHENG, J. A Grid-Based Evolutionary Algorithm for Many-Objective Optimization. *IEEE Trans. Evolutionary Computation 17*, 5 (2013), 721–736.

[200] YAQIN, Z., BEIZHI, L., AND LV, W. Study on job-shop scheduling with multi-objectives based on genetic algorithms. In *Proceedings of 2010 International Conference on Computer Application and System Modeling* (2010), vol. 10, IEEE, pp. V10–294.

[201] YENISEY, M. M., AND YAGMAHAN, B. Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends. *Omega 45* (2014), 119–135.

[202] YUAN, Y., AND XU, H. A memetic algorithm for the multi-objective flexible job shop scheduling problem. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation* (2013), ACM, pp. 559–566.

[203] ZAHIRI, A., AND AZAMATHULLA, H. M. Comparison between linear genetic programming and M5 tree models to predict flow discharge in compound channels. *Neural Computing and Applications 24*, 2 (2014), 413–420.

[204] ZHANG, G., SHAO, X., LI, P., AND GAO, L. An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. *Computers & Industrial Engineering 56*, 4 (2009), 1309–1318.

[205] ZHANG, Q., AND LI, H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *Evolutionary Computation, IEEE Transactions on 11*, 6 (2007), 712–731.

[206] ZHANG, Q., ZHOU, A., ZHAO, S., SUGANTHAN, P. N., LIU, W., AND TIWARI, S. Multiobjective optimization test instances for the CEC 2009 special session and competition. *University of Essex, Colchester, UK and Nanyang technological University, Singapore, special session on performance assessment of multi-objective optimization algorithms, technical report* (2008), 1–30.

[207] ZHANG, R., CHANG, P.-C., SONG, S., AND WU, C. Local search enhanced multi-objective pso algorithm for scheduling textile production processes with environmental considerations. *Applied Soft Computing 61* (2017), 447–467.

[208] ZHANG, X., TIAN, Y., AND JIN, Y. A Knee Point-Driven Evolutionary Algorithm for Many-Objective Optimization. *IEEE Trans. Evolutionary Computation 19*, 6 (2015), 761–776.

[209] ZHAO, F., CHEN, Z., WANG, J., AND ZHANG, C. An improved moea/d for multi-objective job shop scheduling problem. *International Journal of Computer Integrated Manufacturing 30*, 6 (2017), 616–640.

[210] ZHU, G.-Y., DING, C., AND ZHANG, W.-B. Optimal foraging algorithm that incorporates fuzzy relative entropy for solving many-objective permutation flow shop scheduling problems. *IEEE Transactions on Fuzzy Systems* (2020).

[211] ZITZLER, E., LAUMANNS, M., AND THIELE, L. SPEA2: Improving the strength pareto evolutionary algorithm. In *EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems* (2002), pp. 95–100.

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