Survey on Genetic Programming and Machine Learning Techniques for Heuristic Design in Job Shop Scheduling

Fangfang Zhang, *Member, IEEE,* Yi Mei, *Senior Member, IEEE,* Su Nguyen, *Member, IEEE,* and Mengjie Zhang, *Fellow, IEEE*

***Abstract*—Job shop scheduling is a process of optimising the use of limited resources to improve the production efficiency. Job shop scheduling has a wide range of applications such as order picking in the warehouse and vaccine delivery scheduling under a pandemic. In real-world applications, the production environment is often complex due to dynamic events such as job arrivals over time and machine breakdown. Scheduling heuristics, e.g., dispatching rules, have been popularly used to prioritise the candidates such as machines in manufacturing to make good schedules efficiently. Genetic programming, has shown its superiority in learning scheduling heuristics for job shop scheduling automatically due to its flexible representation. This survey firstly provides comprehensive discussions of recent designs of genetic programming algorithms on different types of job shop scheduling. In addition, we notice that in the recent years, a range of machine learning techniques such as feature selection and multitask learning, have been adapted to improve the effectiveness and efficiency of scheduling heuristic design with genetic programming. However, there is no survey to discuss the strengths and weaknesses of these recent approaches. To fill this gap, this paper provides a comprehensive survey on genetic programming and machine learning techniques on automatic scheduling heuristic design for job shop scheduling. In addition, current issues and challenges are discussed to identify promising areas for automatic scheduling heuristic design in the future.**

***Index Terms*—Scheduling Heuristics, Automatic Learning, Genetic Programming, Job Shop Scheduling, Machine Learning.**

I. Introduction

Given a set of machines and jobs, job shop scheduling (JSS) aims to generate schedules for producing jobs (e.g., products and services) to customers to minimise lead time and total costs [1], [2]. JSS is an important combinatorial optimisation problem that has various applications such as goods supply in supermarkets [3], [4] and nursing service scheduling in home healthcare [5], [6]. A good schedule for JSS is a key to increasing the enterprise benefits by improving customer satisfaction and loyalty [7], and helps businesses react quickly to disruptions (e.g., panic buying due to COVID-19 [8], [9]). In reality, JSS is difficult, since the information of dynamic events is not known in advance and multiple decisions must be coordinated [10].

JSS is a typical example of production scheduling, which covers a number of real-world problems such as auto parts processing in car factories [11], and staff scheduling in airports [12] and hospitals [13]. There are mainly three types of methods for handling the JSS problems. *Exact methods* such as dynamic programming [14], have been used to find the optimal solution for JSS. However, they are not efficient and are normally used in small scale and static problems. *Meta-heuristic methods* such as genetic algorithms [15] can handle a wide range of large scale combinatorial problems more efficiently than exact methods However, they face with rescheduling problems and cannot react to dynamic problems in time. Some recent surveys of meta-heuristic methods in JSS can be found [16], [17]. *Scheduling heuristics* such as dispatching rules [18]–[20], e.g., shortest processing time, have been popularly used to optimise either machines or operations as priority functions. However, designing an effective scheduling heuristics needs domain knowledge from experts that is not always available. In addition, manually designed scheduling heuristics are only suitable for the specific scenario they are designed for, but become ineffective in other scenarios. To deal with the above limitations, genetic programming (GP) [21]–[23], as a hyper-heuristic approach, has been popularly used to learn scheduling heuristics automatically [24]. In the past decades, a number of studies have been conducted to improve the performance and (or) efficiency of GP on JSS from different aspects such as advanced genetic operators [25] and parent selection method [26]. More importantly, GP algorithms with machine learning techniques such as ensemble learning [27], surrogate [28], feature selection [29], multitask learning [10], and multi/many-objective optimisation [30], have been established in recent years on handling scheduling heuristic design. This survey will provide a comprehensive study on automatic scheduling heuristic design for JSS with GP from different aspects.

Fig. 1 shows the number of publications of GP for automatically scheduling heuristic design in JSS over recent ten

Manuscript received XXX; revised XXX; accepted XXX. This work is supported in part by the Marsden Fund of New Zealand Government under Contract MFP-VUW1913, and by the MBIE SSIF Fund under Contract VUW RTVU1914. The work of Su Nguyen is funded by Vingroup Innovation Foundation (VINIF) under project code VINIF.2022.DA00183. (Corresponding author: Fangfang Zhang.)

Fangfang Zhang, Yi Mei, and Mengjie Zhang are with the Evolutionary Computation Research Group, School of Engineering and Computer Science, Victoria University of Wellington, Wellington 6140, New Zealand (e-mail: fangfang.zhang, yi.mei, mengjie.zhang@ecs.vuw.ac.nz). Su Nguyen is with the Centre for Data Analytics and Cognition, La Trobe University, Melbourne, VIC 3086, Australia (e-mail: p.nguyen4@latrobe.edu.au).

This article has supplementary downloadable material available at XXX, provided by the authors.

Colour versions of one or more of the figures in this article are available online at XXX.

Digital Object Identifier XXX

This article has been accepted for publication in IEEE Transactions on Evolutionary Computation. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2023.3255246

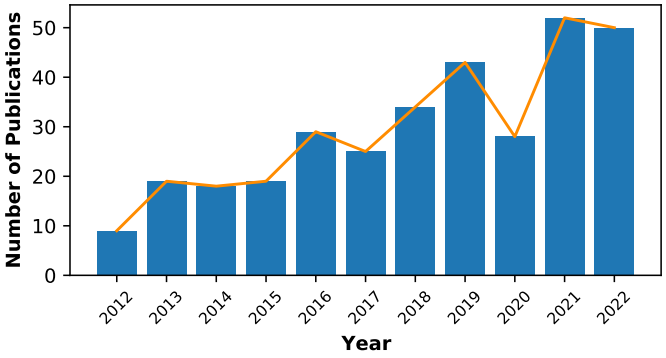

Fig. 1. The number of publications of GP for scheduling (from Web of Science by using keywords "genetic programming" and "scheduling" for topics, in March 2023). The line indicates the trend of the number of publications over the years.

years. Overall, the number of publications has been increasing over time with some fluctuations. Recent reviews on automatic scheduling heuristic design for JSS can be found in [2], [31]. The design choices including the learning methods, representations of priority functions and evaluation of hyper-heuristic approaches, were discussed in 2016 [2]. This study provides us with a high-level picture of automatic scheduling heuristics design with basic GP on limited types of JSS. A unified GP framework was studied for JSS in 2017 [31], which provides details of how to use GP for learning scheduling heuristics in terms of GP basics on general JSS. Over the past years, machine learning has been becoming more and more popular for various tasks [32], and there are an increasing number of studies investigating GP with machine learning techniques such as feature selection and multitask learning to learn scheduling heuristics JSS. More importantly, these methods have achieved great success. However, the existing reviews [2], [31] do not cover these recent studies. A survey of how to use machine learning for combinatorial optimisation was introduced in [33]. However, [33] focuses only on a methodological point of view by introducing how machine learning techniques have been used for general combinatorial optimisation in a high level instead of surveying the state-of-the-art studies in machine learning. Furthermore, the existing surveys [2], [31], [33] are mainly limited to a single type of JSS problems.

A comprehensive survey is worth investigating for promoting and supporting the studies of recent trends for GP in JSS. To fill this gap, we aim to provide a comprehensive survey of the state-of-the-art work with discussions of the open issues and challenges for future work. Different from existing surveys, this survey will investigate the recent GP designs for various types of JSS problems. Furthermore, this paper will focus on how machine learning techniques have been incorporated into GP for automatic scheduling heuristic design. We expect this survey to attract attention from researchers working on GP to address new challenges in JSS and other combinatorial optimisation problems. This survey also aims to push further recent attempts of incorporating machine learning techniques with GP for automatic scheduling heuristic design.

The remainder of this paper is organised as follows. Section II introduces the JSS problems and the main methods for JSS. Section III introduces the recent designs of GP for automatically scheduling heuristic learning in JSS. Typically GP approaches for different types of JSS in Section IV. Section V reviews of GP with machine learning techniques in JSS. The applications of GP for JSS are described in Section VI. Section VII discusses the current issues and challenges followed by the conclusions in Section VIII.

II. Problems and Methods

A. Job Shop Scheduling

In JSS, m machines $\mathcal{M} = \{M_1, M_2, ..., M_m\}$ have to process n jobs $\mathcal{J} = \{J_1, J_2, ..., J_n\}$. Each job J_j has a sequence of operations $\mathcal{O}_j = \{O_{j1}, O_{j2}, ..., O_{jl_j}\}$ that need to be processed one by one, where l_j is the number of operations of job J_j [34], [35]. The finish of all the operations of a job indicates the completion of a job. Below are the main constraints of the JSS problems [36]–[38].

- A machine can only process one operation at a time.
- At any given time, each operation can be handled by only one machine.
- An operation cannot be handled until all of its precedents have been processed.
- Once an operation has been started, the processing cannot be paused or stopped until it has been completed.

Job/machine information availability and the flexibility of machine resources are the main streams in categorising the JSS problems. Correspondingly, the required decisions needed to make in different JSS problems vary.

1) Categories: Based on the *availability of job or machine information*, the JSS problems are generally classified into two groups: *static JSS* and *dynamic JSS*. For static JSS, all the information of jobs and machines is available before generating schedules. For dynamic JSS, not all the information of jobs and machines is available. For example, job information is accessible only after the jobs come to the job shop [39], [40]. This indicates that people only know partial information of jobs to make schedules. Another commonly considered dynamic event is machine breakdown where some machines can break unexpectedly [41]–[43]. Dynamic JSS is harder than static JSS due to the uncertainty brought by dynamic events.

According to the *flexibility of machine resources*, the JSS problems are normally grouped into two categories: *non-flexible JSS* and *flexible JSS*. In non-flexible JSS, each operation can only be processed by one predefined machine, whereas each operation can be processed on more than one machine in flexible JSS [44]–[46]. Flexible JSS is more challenging than non-flexible JSS, since multiple decisions need to be made simultaneously.

Except for them, the JSS problems can be grouped based on specific problem characteristics. For example, some JSS problems consider machine setup times [47], while others treat the setup time as zero [48]. In this survey, we will use the main streams for categories, i.e., static versus dynamic, and non-flexible versus flexible, to introduce the literature wherever needed. Considering that the aim of this paper is on the automatic scheduling heuristic design with GP which

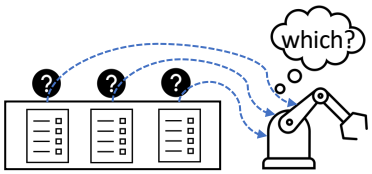

Fig. 2. An example of a sequencing decision situation for an idle machine with three operations in non-flexible JSS.

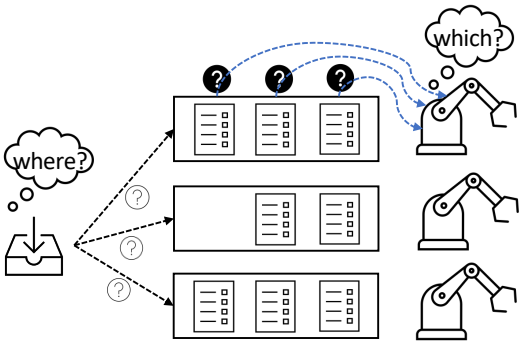

Fig. 3. An example of a routing and a sequencing decision situation with three machines in flexible JSS.

is mainly for dynamic JSS, specifically, three types of JSS, i.e., static JSS, dynamic JSS, and dynamic flexible JSS, are the main focus of this paper.

2) Decision Making: The decision making process varies in different types of JSS, which mainly depends on the flexibility of machines, i.e., non-flexible JSS versus flexible JSS. One only needs to make a *sequencing* decision that selects an operation to be processed next in non-flexible JSS, since where to allocate an operation is already predefined. Fig. 2 shows an example of an operation sequencing decision situation with an idle machine and three operations in JSS, where we need to decide which operation to be processed next by this idle machine. In flexible JSS, two kinds of decisions, i.e., a *routing* decision to allocate operations to machines when an operation becomes ready, i.e., machine assignment, and a sequencing decision to select the next operation to be processed by an idle machine, i.e., operation sequencing. Fig. 3 shows an example of the routing decision in flexible JSS given a ready operation, i.e., the first operation of a new job or an operation whose precedent operation is just completed, we need to decide which machine to process this operation.

Table I shows the characteristics of the information availability and decision requirement in different types of JSS. We can see that static JSS is generally easier than dynamic JSS, since we know all the job information in static JSS. Dynamic flexible JSS is the hardest one with multiple highly interdependent decisions under uncertain environments.

TABLE I
THE CHARACTERISTICS OF THE AVAILABILITY OF INFORMATION AND DECISION REQUIREMENT IN DIFFERENT TYPES OF JSS.

	Problem	Static JSS	Dynamic JSS	Dynamic Flexible JSS
Information	known	√		
	unknown		√	√
Decision	routing			√
	sequencing	√	√	√

B. Methods for Job Shop Scheduling

This section briefly summarises the existing techniques for handling the JSS problems according to the search techniques.

1) Exact Methods: Exact methods are able to find an optimal solution to an optimisation problem [49]. The JSS problems were studied with branch and bound algorithms with 10 jobs and 10 machines in [50], and with a maximal 12 jobs and 3 machines in [51]. The studies with dynamic programming can be found in [14], [52]. These early studies only consider static and small scale instances. However, as problem size increases, it becomes very challenging for exact methods to achieve good solutions efficiently [53].

2) Meta-heuristic Methods: At the early stage, typical heuristic search methods such as tabu search and simulated annealing, were widely studied used to find solutions for static and small scale JSS problems. For example, tabu search was developed for static JSS instances with a maximal number of operations of 30 and machines of 15 [54]. Heuristics methods such as genetic algorithms [55], [56], particle swarm optimisation [57] and their hybrid algorithms [58] can find good enough solutions in reasonable time, which can handle large scale problems well [59]. The basic idea of these methods is to represent the solutions by vectors based individuals, i.e., encoding. Then, the solutions are updated according to the mechanism of corresponding heuristic algorithms. When the stopping criterion is met, the best learned individual is decoded to get the final solution. However, one drawback of these algorithms is that they cannot react to dynamic events efficiently since they face with rescheduling issues with dynamic events.

3) Scheduling Heuristics: Scheduling heuristics such as dispatching rules [60]–[62], as priority functions, have been popularly used to handle JSS problems by prioritising machines and operations, especially for dynamic JSS. An example of a popularly used scheduling heuristic for machine assignment is LWIQ, i.e., least work in the queue, and for job sequencing is FCFS, i.e., first come first serve. There are a number of advantages of using scheduling heuristics. First, with scheduling heuristics, the decision making process in JSS is efficient since we only make decisions at the decision points such as when a new job comes or when a machine becomes idle. Second, the calculations of priority values of machines or operations with scheduling heuristics are efficient, and can react to real-world dynamic events quickly. Third, scheduling heuristics are easy to be used and also can incorporate domain knowledge if we have, such as adding a component into the scheduling heuristics to balance the workload of machines [63]. Last but not least, the priority based decision making nature of scheduling heuristics is more interpretable since we can understand how machines and operations are prioritised by scheduling heuristics (i.e., rules). However, the scheduling heuristics are normally manually designed by experts who have domain knowledge of specific scenarios. The experts are not always able to detect all the attributes of a job shop,

This article has been accepted for publication in IEEE Transactions on Evolutionary Computation. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2023.3255246

especially in complex scenarios. Furthermore, the designed schedules are problem-dependent, and hardly be applied to other scenarios.

4) Hyper-heuristic Methods: Hyper-heuristics methods can automatically learn scheduling heuristics without requiring rich domain knowledge [64]–[68]. Instead of finding solutions for problems directly, hyper-heuristic methods try to find heuristics for generating solutions. This significantly improves the efficiency of handling JSS problems, especially dynamic JSS. Hyper-heuristic methods have been successfully applied to learn scheduling heuristics in JSS. Hyper-heuristic methods can be categorised into two groups, i.e., *heuristic selection* [69] and *heuristic generation* [70]. Heuristic selection aims to find a sequence of heuristics to decide which heuristic to be used in different situations. Heuristic generation aims at generating a comprehensive heuristic that is effective to cover a number of different situations. GP, as a heuristic generation approach, has been widely used to learn scheduling heuristics for JSS [31]. A GP individual can be considered as a priority function where job shop features are its variables. A GP individual handles JSS problems by prioritising machines or operations via calculating the output of the GP individual with feature values based on the job shop status at the decision points discussed in Section II-A2. After the GP training process, the best scheduling heuristic or dispatching rule is obtained. Then, given an unseen JSS instance (either static benchmark JSS instance or dynamic JSS simulation), the learned scheduling heuristic is applied to make scheduling decisions to obtain the corresponding solution (schedule).

Due to its flexible representation, GP can learn scheduling heuristics for JSS automatically without requiring rich domain knowledge. For example, we do not need to define structures (or the shapes) of scheduling heuristics in advance. Note that the optimal structures of scheduling heuristics are normally unknown. In addition, the scheduling heuristics learned by GP are commonly tree-based which have good interpretability. In recent years, researchers have started to use deep reinforcement learning for JSS [71], [72], and deep reinforcement learning algorithms are heuristic selection methods that select heuristics for use in different states/cases [73], [74]. It is noted that GP is the focus in this paper.

The above methods from exact methods to hyper-heuristic methods can be further categorised into two groups, i.e., solution based search approaches that find solutions for specific scheduling problems directly, and heuristic based approaches that learn scheduling heuristics to generate solutions. Exact methods and meta-heuristic search methods belong to the former group. Scheduling heuristics and hyper-heuristic methods belong to the latter group. GP is the focus in this paper.

C. Machine Learning

Machine learning aims to learn generalised models that can find the patterns from training data, and can also perform well on unseen data [75]. Machine learning techniques such as feature selection, transfer learning, and ensemble learning have been widely used in various tasks such as regression and classification tasks [76]. Most current machine learning techniques are used in continuous numeric optimisation problems.

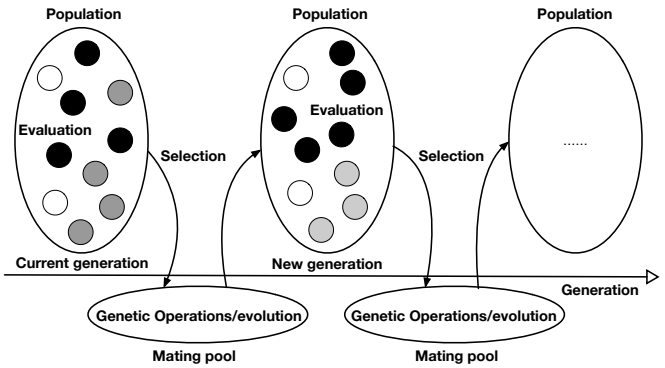

Fig. 4. The evolutionary process of GP.

However, the applications of machine learning to discrete combinatorial optimisation are still very limited. Recent studies have shown a good trend in using machine learning techniques in combinatorial optimisation [33]. However, the studies of machine learning on dynamic combinatorial optimisation is rare. In recent years, the incorporation of machine learning techniques into GP has shown to be promising direction for automatic scheduling heuristic design in JSS, especially in dynamic JSS [77]. For example, there are different features for consideration in the decision making process, however, the importance of them vary. The machine learning technique, i.e., feature selection, can help detect feature importance to make better decisions in this case. It is noted that GP is the focus of this paper, although there are studies of meta-heuristics methods with machine learning techniques, they are beyond the scope of this paper.

D. Detailed Coverage of This Paper

For the rest of this paper, we will introduce GP basics in JSS in Section III. According to two different criteria, GP approaches are classified into different categories. The first criterion is based on problem characteristics, which is the mainstream to classify the JSS problems. Typically GP approaches for different types of JSS including static, dynamic and dynamic flexible JSS, are reviewed in Section IV. The second criterion is based on new techniques. Section V reviews the trendy studies about GP with machine learning techniques including feature selection, surrogate, ensemble learning, multitask learning, and multi-objective learning, for solving JSS. The applications of GP for JSS are described in Section VI.

III. Designs of GP in Job Shop Scheduling

GP, as a hyper-heuristic approach, is the most popularly used hyper-heuristic approach to learn scheduling heuristics for JSS automatically due to its flexible representation [38], [78]–[82]. Fig. 4 shows the evolutionary process of GP. The same as all other evolutionary computation algorithms, GP improves the quality of individuals generation by generation via a loop of individual program evaluation, (parent) selection, and breeding/generation of new individuals. The qualities of individuals in the population vary, i.e., a darker circle indicates a better individual, and we give better individuals more chances to breed new offspring for the next generation.

TABLE II
THE COMMONLY USED TERMINAL SET AND FUNCTION SET OF GP IN JSS.

Categories	**Terminals**	Description
Job	WKR	Median amount of work remaining for a job
	NOR	The number of operations remaining for a job
	W	Weight (importance) of a job
	TIS	Time that a job has been in the job shop
	DD	Due-date of a job
Operation	PT	Operation processing time on a machine
	NPT	Median processing time for the next operation
	OWT	The waiting time of an operation
Machine	NIQ	The number of operations in a machine's queue
	WIQ	The workload of a machine
	MWT	The time of a machine becomes idle
Functions	$+, -, *$, protected $/$, $\max, \min, if$	

A. Representation

How to represent scheduling heuristics with GP is the first step that needs to be considered when using GP to learn scheduling heuristics for JSS.

From the perspective of the components of GP individuals, each GP individual consists of terminals and functions. The problem-dependent terminals provide proper features of a problem for GP to learn from. In JSS, terminals are normally defined by extracting the features of jobs and machines [29], [83], which are shown in Table II. Feature importance varies for generating the routing rule and the sequencing rule [77]. In addition, the feature importance of the routing rule and the sequencing rule also depends on the scenarios, e.g., with different objectives [84]. The functions aim to combine the terminals to GP programs, and arithmetic functions are often used. It is noted that the terminals and functions are not limited to the ones in Table II. Different terminals and functions can be defined depending on the characteristics of examined scenarios. A terminal selection scheme to guarantee the time-invariance throughout the GP process was studied in [83]. The results show that converting terminals to time-invariance ones can achieve better performance than using the original terminals extracted from jobs or operations.

From the perspective of the representative forms of GP individuals, tree-based representation is the most commonly used one to learn scheduling heuristics for JSS. Fig. 5 shows an example of a tree-based scheduling heuristic. This scheduling heuristic can be regarded as a priority function, i.e., NIQ+WIQ*MWT, to generate schedules for JSS. A computational study of representations in GP including decision-tree like representation, arithmetic representation and mixed representation, to learn dispatching rules for JSS was conducted in [85]. The results show that the arithmetic representation, i.e., tree-based representation, is the most effective one among them. Except for tree-based GP, some studies have started to use linear GP for dynamic JSS [34], [86], [87]. In addition, gene expression programming has been used to learn scheduling heuristics with vector based representation which can be interpreted as priority functions for JSS [88]–[90]. Grammar-guided GP was also developed for JSS [85]. Although the representation of GP individuals can be different, all the learned scheduling heuristics work as functionality which can be interpreted as priority functions.

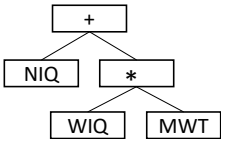

Fig. 5. An example of tree-based GP individual.

B. Evaluation

Evaluation is the process of measuring the quality of GP individuals in JSS. In JSS, the optimised objectives are time-related such as flowtime and tardiness which are commonly used objectives in JSS. The calculations of several commonly used objectives are shown as below [16].

- Max-flowtime: $max_{j=1}^{n}\{C_j - r(j)\}$
- Mean-flowtime: $\frac{\sum_{j=1}^{n}\{C_j - r(j)\}}{n}$
- Mean-weighted-flowtime: $\frac{\sum_{j=1}^{n} w_j * \{C_j - r(j)\}}{n}$
- Max-tardiness: $\max_{j=1}^{n} max\{0, C_j - d_j\}$
- Mean-tardiness: $\frac{\sum_{j=1}^{n} max\{0, C_j - d_j\}}{n}$
- Mean-weighted-tardiness: $\frac{\sum_{j=1}^{n} w_j * max\{0, C_j - d_j\}}{n}$

where C_j represents the completion time of a job J_j, r_j represents the release time of J_j, d_j represents the due date of J_j, and n represents the number of jobs.

*Benchmarks or Training Instances:*Training data are used to learn the scheduling heuristics by measuring the goodness of evolved scheduling heuristics. For static JSS, there are some popularly used benchmarks which are tabular data [91]. For dynamic JSS, simulation is commonly used to mimic the dynamic environments of JSS [92], [93]. The training instances in dynamic JSS are instantiations of the simulation.

Both for static and dynamic JSS, for evaluating the GP individuals, the GP individuals will be used to make routing and (or) sequencing decisions to get the final schedule. According to the obtained schedule, we can know when a job is processed and when it is finished. Thus, the objective values of the schedule can be obtained. The fitness of a GP individual equals to the objective value of the generated schedule by the examined individual. It is noted that for dynamic JSS, normally warm-up stage is used to get a steady-state before the calculations of fitness of GP individuals [94], [95].

C. Parent Selection

Parent selection plays an important role in selecting promising individuals to generate offspring for the next generation [96], [97]. It is an algorithm pressure control mechanism in GP [98]. We expect to use a good set of individuals as parents to generate effective and diverse offspring. Tournament selection is a popularly used parent selection method in GP [99]. There are two main steps of using tournament selection for selecting parents. First, a number of individuals are randomly sampled from the population. Second, the individual with the best fitness among the sampled individuals will be selected as a

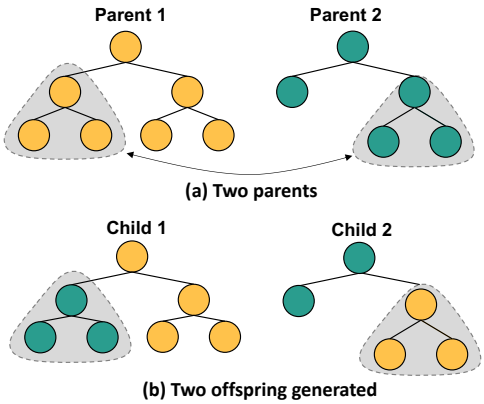

Fig. 6. An example of generating two offspring with two parents via crossover.

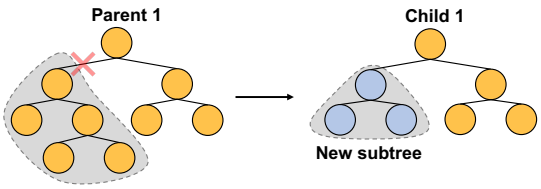

Fig. 7. An example of generating one offspring with a single parent via mutation.

parent. A larger tournament selection size indicates a more greedy parent selection pressure.

There are a number of studies of GP algorithms with more advanced parent selection strategies for JSS. First, the parent selection was realised by defining a new fitness function that takes subtree importance into consideration [100]. Specifically, the subtree importance was combined into the fitness function in a weighted manner in [100]. The individuals with more important subtrees are encouraged to be parents to generate offspring. Second, the parents were selected based on not only their fitness but also diversity [101], [102]. Last but not least, GP with multi-case fitness strategies were studied in [26], [103], in which [26] is a lexicase selection approach. Specifically, instead of evaluating a GP individual on one case instance, the proposed GP algorithm measured the fitness of a GP individual in multiple cases, and decided to choose parents across different cases. The designed algorithm is expected to maintain the diverse individuals that are good in different subsets of cases in the population. A similar idea of keeping the diversity of individuals was studied in [104]. Specifically, the good individuals from different generations with different training instances are archived, and are assigned probabilities of being selected as parents in the current population.

D. Evolution

Three genetic operators, i.e., crossover, mutation, and reproduction, are commonly used for generating offspring in GP [105], [106]. Fig. 6 shows an example of generating two children via crossover by swapping subtrees between parents. Fig. 7 shows an example of generating one child via mutation. We can see that mutation uses one parent to generate offspring by replacing one of its subtrees with a randomly generated new subtree. Reproduction simply copies the selected parent to the next generation. Except for these three genetic operators, the elitism mechanism copies the best several individuals into the next generation directly with the goal of guaranteeing the best individual in the current generation is not worse than that in the previous one in the evolutionary process.

Among all the genetic operators, the crossover is the most important one for GP to generate new offspring in JSS [107]. Traditional improvement on genetic operators of GP mainly has three categories, i.e., adaptive rate for genetic operators to achieve a good balance of exploration and exploitation [108], [109], depth-dependent crossover that limits the choose of selecting subtree at specific depth [110]–[112], and semantic crossover that produce offspring which bias to some defined semantics [113]–[115]. However, these algorithms can only provide a rough guidance/range on choosing the subtrees rather than specifying effective subtree points.

The existing studies on controlling the swap of specific subtrees can be categorised into two groups, i.e., feature frequency based and subtree contribution based. These methods are based on the knowledge that the subtree importance for a GP tree can be different. Removing an important subtree might change the behaviour of a GP tree significantly, which can lead to worse performance [116], [117]. With the feature importance information based on feature frequency, [118] proposed a subtree importance measure by considering the subtrees with more important features are more significant, and gave more chance to remove unimportant subtrees from a GP individual and gave higher probabilities to get important subtrees from the other parent. However, subtree importance information based on the frequency of features is not accurate, since GP individuals may contain redundant subtrees. To this end, [25] measured the subtree importance by comparing the behaviour of the subtree with the behaviour of the whole tree, where the behaviour is represented by phenotypic characterisation [37]. If the absolute value of the correlation of their behaviours is close to 1 or 0, this indicates the examined subtree is important or unimportant for the corresponding GP individual. Then, for operating the crossover, if the offspring are generated by removing an unimportant subtree by replacing an important subtree from the other parent, the generated offspring are more likely to have good quality.

We can see that instead of enhancing the effectiveness of GP via crossover by giving high-level guidance such as adaptive operator rate and defining subtree depth range, we can work on the low-level strategy to decide which subtree to choose specifically. The key is how to measure the subtree importance for corresponding problems. The idea of improving offspring generation efficiency with genetic operators is generic and can be extended to other problems. In addition, the subtree importance information can be used for other functions such as implying GP individuals by removing unimportant subtrees.

In summary, from this section, we can see that GP can automatically learn scheduling heuristics for JSS with terminals and functions. Different representations of GP can be used to learn scheduling heuristics in JSS. Enhanced parent selection and genetic operators strategies can improve the effectiveness of GP in JSS.

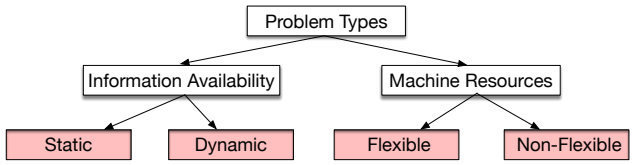

Fig. 8. Criteria for categorising JSS problems.

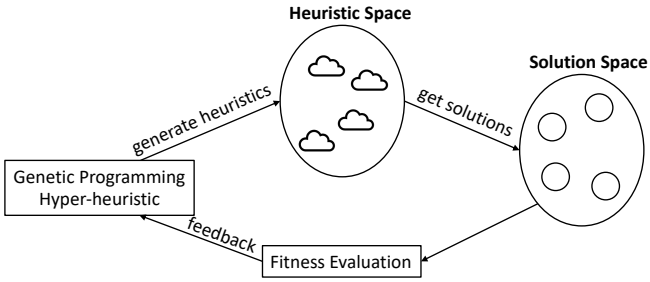

Fig. 9. Learning in heuristic space with GP.

IV. GP for Different Job Shop Scheduling

We group the algorithms for JSS according to the types of JSS by considering the available information and machine resources as shown in Fig. 8.

A. GP for Static JSS

Although GP is widely used to learn scheduling heuristics in dynamic JSS, it is applicable for static JSS [85], [119]. Static flexible JSS was also studied with GP in [120]. The results show that GP can find better solutions than the manually designed rules. The learning processes of GP on training data to handle either the static or dynamic JSS problems are the same. However, the goals of using GP for JSS can be different. The GP algorithms on static JSS can be treated as a learning approach [85], [119], to learn the scheduling heuristics. Then, the learned scheduling heuristics will be applied to test instances to measure their generalisation. GP can also be treated as an optimisation approach to get the solutions/schedules for each problem directly [120]. In other words, there is no training and test process.

Either taking GP as a learning or an optimisation approach to static JSS, GP searches in heuristic space as a hyper-heuristic approach, and the evaluation of GP individuals is the same. Fig. 9 shows a flowchart of using GP for JSS with a focus on the individual evaluation process. We can see that for fitness evaluation of the learned heuristics, we need to convert the heuristics to solutions to get objectives. In other words, the quality of a learned heuristic is not possible to be measured directly, and we need to measure the quality of heuristics by examining its corresponding solution. Since all the information for making schedules in static JSS is known in advance, either taking GP as an optimisation or a learning approach is reasonable, since the generalisation ability of learned scheduling heuristics is not necessary a requirement.

B. GP for dynamic JSS

Different from static JSS, the information of jobs or machines is not totally known in advance in dynamic JSS [121]–[123]. With the same types of dynamic events such as job arrival and machine breakdown, different instances may have different characterisations due to the difference between event distributions. Thus, the decisions between different instances with the same types of dynamic events can be different. Normally, we take GP as a learning algorithm, where the learned scheduling heuristics are expected to have good generalisation to handle a family of unseen instances well.

In traditional non-flexible dynamic JSS, an operation can be processed on a predefined machine. The decision we need to make is operation sequencing for an idle machine. Machines are the main resources in the JSS system that need to be optimised for production [124], [125]. In this paper, we categorise the dynamic JSS into two groups according to the number of machines in the JSS system.

1) Single Machine: For the job shop with a single machine, we only need to prioritise operations on one machine. Normally one dispatching rule is used for this purpose. The effectiveness of GP on learning such a dispatching rule for dynamic JSS with a single machine was investigated in [126]–[132]. These studies with a single machine were mainly conducted at the very early stage of JSS, and are out of sight gradually due to their less practicality.

2) Multiple Machines: It is more common that there are more than one machine in the job shop, and we need to prioritise operations of multiple machines, especially in modern manufacturing [133], [134]. From the perspective of the number of used dispatching rules, there are typically two ways to handle dynamic JSS with multiple machines. The first one is to use a global scheduling heuristic to prioritise operations for all machines [37], [135]–[137]. The second one to use multiple dispatching rules for machines, and each scheduling heuristic is corresponding to a single machine [138], [139]. The first approach that learns a dispatching rule to prioritise operations for all machines is the mainstream in current research. It has the following advantages. First, it improves the interpretability of scheduling heuristics, since explaining more rules simultaneously is more complex than one rule. Second, it simplifies the learning process of GP since GP can focus on learning one comprehensive rule rather than more rules simultaneously which requires to consider the interaction between the rules.

C. GP for Dynamic Flexible JSS

The special characteristic of dynamic flexible JSS is that each operation can be processed on more than one machine [140]–[142], and machine assignment and operation sequencing decisions need to be made simultaneously [143]. Dynamic flexible JSS is more complex than other types of JSS due to the complicated interactions between the two decisions. Dynamic flexible JSS has been attracting more and more attention from researchers due to its practical value [144]–[146].

From the respective of GP basics, how to define proper terminals for learning both the routing rule for machine assignment and the sequencing rule for operation sequencing is important for the success of GP in dynamic flexible JSS. The terminal setting investigated in [88] shows that the combinations of job and operation based terminals are important for

learning scheduling heuristics in dynamic flexible JSS. However, [88] only aims to investigate the terminals for learning the sequencing rule, and the routing rule is a fixed manually designed rule. The terminal importance for the routing rule and the sequencing rule were studied in [77], [84], [118], which shows that terminals importance varies for different rules, and also differ for the same type of rules in different scenarios.

From the respective of needed rules in JSS, traditional GP can learn one rule straightforwardly. However, in dynamic flexible JSS problems, more than one rule is needed, i.e., routing rule and sequencing rule need to be learned simultaneously. Earlier studies normally only learn one rule and fix the other rule [147], which are not effective. GP with cooperative coevolution was used to learn the routing rule and the sequencing rule simultaneously in [120]. Simply speaking, two subpopulations are used to learn two rules separately. When evaluating routing/sequencing rules in one subpopulation, the best sequencing/routing rule from the other subpopulation is used together to measure the performance of the rules. Multi-tree representation was proposed to learn the routing rule and the sequencing rule in [148], [149]. Specifically, each individual contains two trees, one for the routing rule and the other for the sequencing rule. More details of comparing different ways to represent more than one rules simultaneously can be found in [150]. The existing study has shown that the importance of the routing rule and the sequencing rule in dynamic flexible JSS is different [151]. A computational resources allocation strategy was proposed to give the routing rule and the sequencing rule different resources which were represented by the number of individuals according to their contributions to fitness improvement in [151]. The results show that the proposed algorithm that emphases more on the important rules, i.e., routing rules, is more effective than the algorithms that ignore the importance of rules.

From the respective of search, it is challenging to guide the search by handling both the routing rule and the sequencing rule in dynamic flexible JSS. A growing neural gas and principal component analysis were applied to generate and update the map of explored areas based on the phenotypic characteristics of evolved heuristics efficiently in [152], [153]. The results show that the developed algorithm can get a good balance between exploration and exploitation.

From this section, we can see that the design of GP for JSS is problem-dependent, and there needs different concerns. In the next section, we will introduce GP in JSS with a focus on machine learning techniques.

V. Machine Learning Techniques for GP in Job Shop Scheduling

There are a number of attempts at incorporating machine learning techniques for JSS in recent years as shown in Fig. 10. As machine learning techniques, feature selection and surrogate have been studied in JSS. From the learning paradigms of machine learning, ensemble learning, multitask learning, and multi-objective learning have been investigated. Although these machine learning techniques have been successfully incorporated with GP for JSS, there is still a lot of potential for working further in these directions.

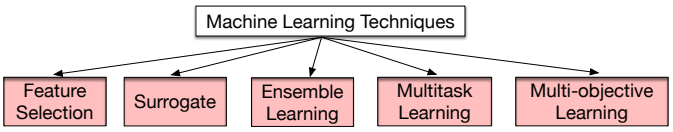

Fig. 10. Different machine learning techniques with GP for JSS.

A. Feature Selection

Feature selection is an important machine learning task that aims to select a small but complementary feature subset for learning models for problems [154]. The goal of the traditional feature selection is to select a small subset of features but to get a good algorithm performance such as high accuracy or low error for classification [155]. Unlike traditional machine learning tasks where features are available as in the tabular data, the features in JSS are extracted based on specific JSS tasks. In theory, there can be an infinite number of features for GP in JSS based on the characteristics of JSS. In reality, however, it is not possible for people to extract all features. Instead, people normally extract features based on domain knowledge in specific decision scenarios. Compared with traditional feature selection, the feature selection is more challenging in GP for JSS.

First, the feature importance may vary in different JSS scenarios, and we need feature selection approaches to detect important features for scheduling heuristic design automatically. Taking the sequencing rule as an example, PT (processing time of operations) is an important feature in optimising mean-flowtime, whereas, W (job importance) is an important feature in optimising mean-weighted-flowtime [84]. Second, the important features for learning the routing rule and the sequencing rule are different. For instance, MWT (machine waiting time) is an important feature for learning routing rules, however, this is not the case in learning sequencing rules [118]. Last but not least, the same scheduling heuristics can be represented in different ways. For example, GP can represent x + 1 as x + x / x. In other words, although with the same number of features, the possible combinations of features are larger than the traditional feature selection.

Although GP can inherently select important features to learn effective programs, the accuracy of its feature selection is not sufficient. For example, the redundant subtrees in a GP individual can lead to the inaccuracy of feature selection. Effective feature selection GP algorithms are worth investigating to provide a good understanding of the feature importance of learning the routing rule and the sequencing rule, and the feature importance in different scenarios.

Feature selection has been proven to be effective for JSS. Different from the vector-based representation in the traditional machine learning tasks, it is more difficult to measure feature importance in tree-based representation. From the literature, there are two kinds of approaches for this purpose.

1) Frequency Based: A simple way to measure the feature importance is to observe the frequencies of features of good GP individuals [84], [118]. This is based on the assumption that GP can automatically detect important features for learning models. If one feature appears more in the individuals

with good quality, the feature is more likely to be important. A similar idea is to calculate the weights/frequency of features in good GP individuals, and use an activation probability function as a threshold to decide whether a feature is important or not [29], [156], [157]. These kinds of approaches cannot measure the feature importance accurately, since GP individuals may contain redundant features.

2) Contribution Based: By considering the contributions of features to the fitness of GP individuals in JSS, feature selection was firstly studied for dynamic JSS in [158]. Simply speaking, to measure the importance of a feature, a feature is replaced by one from a GP individual. The difference of fitness between the newly obtained individual and the original individual can be recognised. After removing the feature, if the fitness of a GP individual becomes worse, then the feature is important. Otherwise, the feature is not important. This approach has also been successfully used in multi-objective optimisation [159]. However, one drawback of this algorithm is that it is time-consuming due to the re-evaluation of newly obtained individuals for measuring the feature importance. To handle this issue, an efficient feature selection algorithm was designed with a niching and surrogate method for dynamic JSS [160]. A niching method was used to get a number of diverse individuals for examining the importance of features. A surrogate method was proposed to reduce the individual evaluation time for feature selection. Specifically, a surrogate model was built by simplifying the original simulation to a smaller simulation with less number of jobs. A new simulation was used to evaluate the individuals for feature selection. After the feature selection stage, the selected features were used to initialise the GP population and start a GP evolutionary process to learn scheduling heuristics for JSS. This approach was also extended to GP with multi-tree representation for dynamic flexible job shop scheduling in [161]. However, the learned scheduling heuristics in the feature selection stage of these mentioned approaches above are completely ignored, and the algorithm normally starts with a population that does not contain good individuals.

To this end, a two-stage GP feature selection algorithm was designed to utilise the learned knowledge from the feature selection stage [162]. In [162], instead of re-initialising the population with selected features only, the two-stage mechanism takes the last population of the feature selection process (stage 1) as the start of a new GP evolutionary process for learning scheduling heuristics (stage 2). The selected features were also used for the mutation operator. However, the scheduling heuristics still contain unselected features, which can affect the interpretability of learned scheduling heuristics. If only using selected features in stage 2 by directing removing unselected features, the performance of the algorithm will become worse. Based on [162], to learn scheduling heuristics only with selected features without deteriorating the performance, a behaviour mimic strategy was proposed in [77] to adapt the individuals from the last population of stage 1 to individuals only with selected features. The results show that the proposed algorithm can achieve effective and smaller scheduling heuristics.

In summary, the GP with feature selection approaches, especially those based on feature contribution, have achieved some success. However, the ways to measure the contributions of features in GP individuals are still narrow. More effective and diverse ways of measuring the feature importance accurately are still needed. For example, the interactions between features are also important for measuring feature importance. In addition, due to the flexible representation, GP can perform feature construction to create new high-level features [163].

B. Surrogate

Surrogate models have been widely used in evolutionary computation to reduce the training time of the algorithms [164]. The commonly used surrogate building techniques are radial basis function [165], [166] and Gaussian process [167]. Most of the existing studies use tabular data to train the surrogate model. However, for GP on dynamic JSS including dynamic flexible JSS, we cannot build surrogate models directly with tabular data as in traditional machine learning tasks. We only have GP individuals and the simulation for measuring the qualities of GP individuals. The existing studies of surrogate assisted GP in JSS can be categorised into two groups, i.e., phenotypic characterisation based and simplified model based.

1) Phenotypic Characterisation Based Surrogate: Phenotypic characterisation was proposed to convert GP individuals/trees to vectors [37] by measuring the behaviour of GP individuals in routing and (or) sequencing decision situations. Phenotypic characterisation consists of the ranks of machines and (or) operations. If two GP individuals have similar phenotypic characterisations, their fitness values are more likely to be similar due to their similar behaviours. A K nearest neighbour (KNN) based surrogate was built with the phenotypic characterisations and the fitness values of real-evaluated GP individuals for dynamic JSS. The fitness of newly generated individuals was estimated by finding the most similar individuals of the samples in the KNN model. The results show that the developed KNN-based surrogate can help preselect good individuals from a large number of generated offspring to the next generation efficiently. The KNN-based surrogate was also successfully used in multitask JSS to allocate individuals to specific tasks properly in [168], and dynamic flexible JSS [104]. In addition, the developed KNN based surrogate [37] was further investigated with different settings in [169]. As we know, a good KNN-based surrogate should have an enough and a diverse number of samples for estimating the fitness of different individuals effectively. To further improve the KNN-based surrogate, an instance rotation based surrogate was developed to improve the accuracy of surrogate model by increasing the number of surrogate samples which was realised by adapting the individual information properly from the previous generation as KNN-based surrogate samples [170].

2) Simplified Model Based Surrogate: Based on the idea of simplification, a surrogate model was built by reducing the number of jobs and machines of the simulation for fitness evaluations of GP in dynamic JSS [171]. The results show that the built surrogate named half-shop simulation

that used half number of the original number of jobs and machines, achieves good performance. This approach was also successfully adapted for dynamic flexible JSS in [20], [172]. Inspired by the study in [171], an adaptive surrogate strategy that increases the fidelities of simulation models along with generations were developed and applied at different generations in dynamic flexible JSS [173]. The results show that the developed algorithm can achieve comparable performance without a shorter training time. In addition, a multi-fidelity based surrogate was designed in [174] where there is more than one surrogate, and they can learn from each other in the evolutionary process of GP. The learning between multiple surrogates is realised by the knowledge transfer via the crossover operator. The results show that the proposed multi-fidelity based surrogate algorithm has a better converge speed and can get better performance than [37] and [171] with the same training time.

Both phenotypic characterisation and simplified model based surrogates are easy to implement. The advantage of phenotypic characterisation based surrogate is that it is computational cheaper than simplified model based surrogate [174]. A comparison of the above surrogate-assisted algorithms was conducted on multi-objective dynamic JSS, and interesting readers can refer to [28]. Although the surrogate building has been established well, almost all of the existing studies only use a surrogate to preselect individuals for the next generation for improving the effectiveness of GP in JSS. The studies that use the surrogate technique to reduce the training time, i.e., normally reducing the time of individual evaluations, is still limited. This is an important research topic, especially when the problem scales are large, and the individual evaluations are computationally expensive.

C. Ensemble Learning

Ensemble learning is a popular machine learning approach that seeks better performance by considering the outputs of multiple complementary models [175]–[177]. The three main classes of ensemble learning methods are bagging, stacking, and boosting [178]. In this survey, we group the existing studies of GP in JSS into two groups, i.e., same instance based and different instances based, according to the ways to build the elements of ensembles. It is noted that an instance in static JSS is a fixed training example. In dynamic JSS, an instance is an instantiation of the simulation, and normally different instances are used at different generations to improve the generalisation of learned scheduling heuristics [31].

1) Same Instance Based: There are a number of approaches that extract learned scheduling heuristics with the same training instance to form ensembles. An ensemble GP algorithm was developed to improve the robustness of learned scheduling heuristics in static JSS problems [179]. Specifically, the population is divided into different subpopulations to learn scheduling heuristics separately. Instead of using one scheduling heuristic for decision making, a number of rules are used to prioritise the operations, and the operation that received the most votes is selected to be processed next. The results show that the proposed ensemble algorithm achieves better performance than the single rule based GP algorithm. This approach was further extended to dynamic JSS in [180]. An investigation of ensemble combination schemes including majority voting, linear combination, weighted majority voting and weighted linear combination, was conducted for GP in dynamic JSS [181]. The results show that linear combination is generally better for the dynamic JSS problem than the other investigated combination schemes. Four different ensemble learning approaches, i.e., simple ensemble combination, BagGP, BoostGP and cooperative coevolution, were studied in [182]. The scheduling heuristics learned by different approaches are combined by a simple sum and vote. The results show that the effectiveness of the vote and sum combination methods are ensemble learning algorithm dependent. In addition, the ensemble sizes heavily depend on the used ensemble learning approaches and the optimised objectives. More analyses of the effects of the ways to create ensembles, the sizes of learned scheduling heuristics, and different ways of selecting the scheduling heuristics for forming the ensembles, were conducted in [183]. A hybrid ensemble algorithm with GP and the genetic algorithm was studied in [184]. GP was used to learn scheduling heuristics, and genetic algorithm was applied to evolve good ensembles of a given maximum size. Instead of working for single objective optimisation, the ensemble GP algorithm was also investigated in multi-objective problems in [185]. The results show that ensemble learning can also perform well in multi-objective JSS optimisation. In addition, the niching technique was applied to learn a diverse set of scheduling heuristics for dynamic JSS [186].

2) Different Instances Based: By taking the difference of problem instances into consideration, an ensemble GP algorithm was proposed to learn a set of scheduling heuristics, and each generates schedules with a goal of distincting regions of the instance space [187]. The results show that the proposed ensemble algorithm performs better than compared algorithms including [179]. Scheduling heuristics were applied independently at each decision point to create a simulation of the schedule for all currently released jobs [27]. Based on these simulations, the scheduling heuristic which makes the best decision will be applied. The results show that the proposed ensembles can easily outperform the approaches that use a single scheduling heuristic. A similar idea can also be found in [188].

In summary, ensemble learning has been developed on GP for JSS, for both traditional static and dynamic JSS. Most of the existing studies focus on extracting ensembles from the same instance by grouping GP individuals in one population or utilising subpopulations, normally at the same generation. Extracting ensembles from different instances is more likely to get diverse ensembles but with less work than generating ensembles from the same instance. In terms of the ways to use generated ensembles, most current approaches simply apply traditional voting or combination methods. In addition, to the best of our knowledge, there has been no study of ensemble learning in dynamic flexible JSS. It is more challenging to build ensembles for dynamic flexible JSS, since both the routing rule and the sequencing rule are needed. For example, one issue is how to measure the diversity of ensembles with two rules. We can see that ensemble learning in GP for JSS is

still in an early stage, and more advanced ensemble techniques with GP are worth investigating.

D. Multitask Learning

Unlike traditional optimisation or learning problems that handle one task at a time, multitask learning aims to handle multiple tasks simultaneously [189], [190]. Under a unified search space, the success of multitask learning lies on the knowledge sharing among tasks during the evolutionary process [191]–[193]. From the existing literature, most of the existing multitask learning algorithms are incorporated with vector-based representation [194], [195]. However, for GP in JSS, the scheduling heuristics in the search space have tree-based representation. In addition, although multitask learning has been widely studied in continuous numeric optimisation problems [196]–[199], its studies on discrete combination problems such as JSS are still limited.

Based on one population, a GP multitask algorithm was designed for handling dynamic scheduling tasks with machine breakdown and job arrival events by learning a generalist and multiple specialist rules [200]. Specifically, this study developed a niched GP approach that incorporates multitask to simultaneously evolve multiple rules that can effectively cope with different machine breakdown scenarios. A multi-population based multitask learning framework was firstly studied on GP in dynamic scheduling problems in [201]. Specifically, each subpopulation was used to handle one dynamic scheduling task, and knowledge sharing between tasks is realised by the crossover operator. Multipopulation based multitask GP in [201] was further studied comprehensively with homogeneous and heterogeneous multitask scenarios generated based on objectives and utilisation levels in [10]. The developed algorithms were tested on different dynamic scheduling scenarios, i.e., homogeneous and heterogeneous multitask scenarios. The results verify the effectiveness of the developed algorithms. This algorithm was further investigated on multi-objective dynamic flexible JSS problems [106]. The results show that multi-population based multitask framework is a better option than single population based multitask framework to learn scheduling heuristics for JSS. Single population and multi-population based multitask linear GP algorithms were also investigated on dynamic JSS [86]. The results suggest using multi-population based multitask GP in JSS, which is consistent with findings in [10], [106].

By focusing on the way of knowledge sharing among tasks, a surrogate-assisted multitask GP algorithm was developed in [168]. Instead of using crossover to realise the knowledge sharing, all the generated offspring for each task from different subpopulations were put into a pool and reallocated for each task. Specifically, all individuals in the pool were reevaluated for each task with the KNN-based surrogate efficiently. The individuals were ranked based on their estimated fitness, and the top ranked individuals for tasks were allocated to the corresponding tasks. Thus, knowledge sharing was realised by allocating individuals generated for a task to another task. The results show that the surrogate-assisted multitask GP has successfully helped the knowledge sharing between tasks by reallocating individuals between tasks effectively and efficiently.

By considering the relatedness of tasks, [202] developed a multitask GP algorithm that can adaptively select a proper task to learn knowledge from. The relatedness measure was developed with the Kullback–Leibler divergence technique based on the GP individuals' information represented by phenotypic characterisations. The results show that the developed GP algorithm can automatically detect important tasks to share knowledge to achieve good performance.

In summary, the foundation of multitask GP has been well established in dynamic JSS and dynamic flexible JSS. First, the way to generate benchmark multitask scenarios using objectives and utilisation levels, has been established, which opens the door for future studies. The effectiveness of multi-population based multitask GP algorithm is also verified, which provides a base for researchers to work with. Some important directions in multitask learning such as the effectiveness of knowledge sharing and the relatedness between tasks have been studied. There are also some other issues that should be addressed. For example, although we know the effectiveness of a knowledge sharing mechanism, it is still not clear what is transferred, and how it contributes to the improvement of performance.

E. Multi-objective Learning

Multi-objective learning is to learn a Pareto front of solutions for handling conflicting objectives [203], [204]. Dominance relation based multi-objective approaches such as NSGA-II [205] and decomposition based multi-objective approaches such as MOEA/D [206], have been popularly used in multi-objective optimisation. In JSS, there are requirements for dealing with conflicting objectives. One provider may want to minimise the maximal tardiness that can deliver products to customers on time. One provider may prefer to minimise mean tardiness to reduce the overall production cost. The objectives of minimising the maximal tardiness and mean tardiness are partially conflicting tasks [207].

1) Traditional Multi-objective Learning: Multi-objective GP was studied for static flexible JSS by weighting objectives with different weights in [208]. The results show that the proposed multi-objective GP algorithm performs better than the manually designed rules, which shows the effectiveness of multi-objective GP algorithm. However, only the sequencing rule for operation sequencing was evolved and a fixed routing rule was used for machine assignment. Multi-objective GP was investigated to explore the potential of learning the Pareto front of rules in dynamic JSS [209] considering job arrivals over time. The results show that multi-objective GP can learn a good and diverse set of rules for dynamic JSS. Incorporating the mechanism of NSGA-II into GP, a multi-objective GP algorithm showed its effectiveness in dynamic JSS with machine breakdown [210]. Island models towards parallel GP for dynamic JSS were studied in [211]. The results show that the proposed algorithm could outperform some general-purpose multi-objective optimisation methods, including the GP algorithms with the mechanism of NSGA-II or SPEA2.

A new diversity metric to measure the distance between GP individuals and the best rule at the current generation was proposed for multi-objective GP with the selection mechanism of NSGA-II in [212]. The results show that the learned scheduling heuristics have a high diversity.

2) Integrated Problem Based Multi-objective Learning: Considering an integrated problem of learning dispatching rules and due-date assignment rules in JSS simultaneously, a diversified multi-objective cooperative evolution GP algorithm was developed with a multi-population framework in [30], [213]. One subpopulation was used to learn the dispatching rules, and the other was designed for evolving the due-date assignment rules. The novelty of this work is the representation and cooperative evolutionary search approaches for handling multiple scheduling heuristics. This algorithm with the co-operative evolutionary mechanism was adapted to dynamic flexible JSS [214] where machine assignment and operation sequencing decisions are needed to be learned simultaneously. The results show that the learned rules are better than the manually rules from the literature. The approach in [214] was further extended with a surrogate technique to improve the individual evaluation efficiency [172]. However, the proposed algorithms in [172], [214] were only compared with the manually designed rules, and there are no comparisons with other algorithms such as GP-based algorithms. Multi-objective GP was studied on dynamic flexible JSS with multi-tree representation in [207]. The results show that the proposed GP algorithm with non-dominated sorting strategy and multi-tree representation for handling machine assignment and operation sequencing decisions performs the best among the compared algorithm. Considering the integration of order acceptance and scheduling problems, a two-stage learning/optimising system was proposed [215]. The novelty of this work is the use of GP to evolve a set of scheduling rules that can be reused to initialise populations of a multi-objective GP algorithm.

Besides the above two main categories, in terms of algorithms, a hybrid algorithm that combines GP and NSGA-III was developed for handling more than three conflicting objectives in [216]. Specifically, the selection scheme of NSGA-III was incorporated into GP to learn scheduling heuristics. To handle the discrete and non-uniform Pareto front issue in multi-objective JSS, a new reference point adaptation mechanism has been proposed in [217] for many-objective JSS. Multi-objective GP with local search [217] was also used to improve the performance of learning scheduling heuristics.

To sum up, multi-objective approaches are a desire for GP to learn scheduling heuristics for JSS, and multi-objective GP has been investigated in different types of JSS problems. Although there are a number of multi-objective GP learning algorithms in JSS, there is a lack of multi-objective GP algorithms with novelties in the key mechanism such as the improvement of dominance relation of individuals. In addition, to the best of our knowledge, the studies of GP with the mechanism of MOEA/D have not been investigated yet. Developing novel multi-objective GP algorithms, particularly the novel search mechanisms, dominance relation and decomposition strategies, are still open issues.

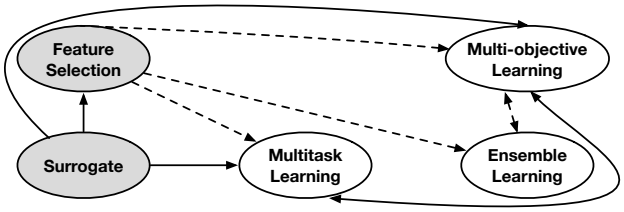

Fig. 11. Connections of machine learning techniques with GP in JSS, where solid and dotted line indicate existing and potential connections, respectively.

F. Other Techniques with GP

There are also some studies that adapt other techniques for GP, or using GP to assist other algorithms, in JSS. GP was used to learn efficient variable selectors to enhance the search mechanism in constraint programming [218]. The proposed algorithm was investigated on the JSS problems, and the results show that the proposed algorithm achieves better performance than the constraint programming with default settings. This work shows the potential of using other techniques for assisting GP for JSS. GP was combined with particle swarm optimisation for JSS in [219]. Specifically, the learned scheduling heuristics were incorporated into particle swarm optimisation by generating the initial global best position that the particles could converge to. In other words, GP is used as an initialisation approach for particle swarm optimisation.

G. Discussions

All the mentioned machine learning techniques are not necessary to be used independently. They can work together to promote the performance of GP algorithms. Fig. 11 shows the connections of introduced machine learning techniques above with GP in JSS. We can see that the surrogate technique has been used to help extra information for feature selection [77], [162], and knowledge sharing in multitask learning [168], and improve the effectiveness of multi-objective GP learning [220]. Multitask multi-objective GP in JSS has been studied in [106].

We can also see some potential connections. For example, feature selection can detect important features between different tasks, and we can use this information to measure the relatedness between tasks in multitask learning. Based on the feature characteristics obtained by feature selection, we can also learn ensembles with ensemble learning by using different groups of features to learn good and diverse scheduling heuristics in JSS. Furthermore, the feature importance of scheduling heuristics in multi-objective learning that needs to handle more than one objective is more likely to be different from single objective optimisation. Utilising this information can help find a better Pareto front by keeping individuals with diverse features. Last but not least, ensemble learning for multi-objective learning in JSS with GP has not been explored, and it is worth exploring in the near future.

VI. Applications

GP for scheduling has been widely used in many real-world applications. Table III shows a number of example applications of GP in scheduling problems. We can see that GP based

TABLE III
APPLICATIONS.

Areas	**Applications**	**References**
(1)	Workflow scheduling in cloud	[221], [222]
	Scheduling in networks	[223]–[225]
	Distributed multiprocessor scheduling	[226]
(2)	Semiconductor manufacturing	[227], [228]
	Steel production scheduling	[229]
	Crude oil scheduling	[230], [231]
	Steelmaking and continuous casting	[232]
	Textile industry	[233]
	Project scheduling problem	[229], [234]–[240]
	Superblock scheduling	[241]
	Container terminal truck dispatching	[242]
(3)	Ambulance Dispatching	[243]
(4)	Earth observing satellite scheduling	[244]

scheduling has been widely applied to various areas. Generally, the main applications can be grouped into four categories: (1) Service management including workflow scheduling in cloud, scheduling in networks, and multiprocessor scheduling, (2) Manufacturing including semiconductor manufacturing and steel production scheduling, (3) Medical applications such as ambulance dispatching, and (4) Astronomy geography including observing satellite scheduling. Among these areas, the GP algorithms on scheduling in manufacturing are the most popular one, and have been widely applied in different areas.

All the above areas are important for our society's development or play significant roles in everyone's daily life. Although not applied yet, GP with machine learning for scheduling has great potential to be applied to logistic scheduling, supply chain scheduling, transportation scheduling and routing under dynamic and uncertain environments, also in primary industry and open ocean industry climate change in the future.

VII. ISSUES AND CHALLENGES

A. Interpretability

Interpretability is a hot topic in machine learning [245]. This is especially true for real-world scheduling applications, since we need to build trust for customers and make people understand how the decisions are made [246]. For example, the operators prefer to know how reasonable the schedules are generated with scheduling heuristics that they can rely on in mobile production manufacturing [247]. Feature selection methods have been used to learn scheduling heuristics with smaller sizes and smaller number of unique features [77]. In addition, from the algorithm design point of view, we need to understand the learning process of GP for JSS rather than regarding the GP algorithm as a black box.

1) Learned Scheduling Heuristics: In terms of the structures of scheduling heuristics, although the tree based priority functions obtained by GP are easier to be interpreted compared with more complex learning algorithms such as neural networks with a large number of layers and weights, it is still challenging to fully understand the decision making of the rules. For example, by looking at the structure of a learned rule, it is not easy to explain how and why the features and functions are combined in that way.

The first question is how to quantify the interpretability of learned scheduling heuristics. Current studies mainly use the sizes of the scheduling heuristics, e.g., the number of nodes for this purpose [77]. However, it is not necessary that a smaller scheduling heuristic is more interpretable. The interpretability of scheduling heuristics is also related to the ways of combinations of terminals and functions.

There are a few studies investigating the scheduling heuristics by looking at the feature importance [84], [118]. This can provide a high level understanding of scheduling heuristics in terms of the needed components. However, it is still not clear why they are combined in this way. To handle this issue, a dimensionally aware GP algorithm was proposed to improve the interpretability of scheduling heuristics by dividing terminals into tree groups, i.e., time-related, count-related and weight and constrainting the terminal combinations within each group [114]. The results show that the proposed algorithm can achieve comparable results as the baseline GP but has better interpretability. However, this algorithm constrains the search space of GP, and this explains why the performance of the scheduling heuristics is not improved.

Another way to explain the learned scheduling heuristics is to look at their behaviour. As we mentioned earlier, phenotypic characterisation is a commonly used technique to represent the behaviour of scheduling heuristics [37]. The behaviour of the scheduling heuristics was investigated by comparing it with the manually designed rules. Specifically, different manually designed rules are sampled to see how much the learned scheduling heuristic are similar to them [213]. One issue of this approach is that phenotypic characterisation in existing studies is based on randomly sampled decision situations which might not be representative enough. However, how to extract the characteristics of decision situations, and how to select a good set of decision situations for measuring the phenotypic characterisation is non-trivial.

2) Evolutionary Process: Understanding the evolutionary process of GP can benefit us in three aspects. First, it turns the black box learning into a white/grey box learning to make a better understanding of GP in JSS. Second, it provides a better understanding of how the scheduling heuristics come out. Third, it provides a chance for people to interact with GP such as guiding the search direction. Visualisation is a good way to show the working mechanism of an algorithm [248].

A people-centric GP system was studied to intervene and embed people's knowledge and preferences in the evolutionary process, and guide GP towards the desired solutions [249]. A mapping technique was used to incrementally monitor the evolutionary process. The results show that the proposed system outperforms the existing algorithms for evolving scheduling heuristics in terms of scheduling performance and heuristic sizes. Dimensionality reduction technique and growing neural gas were used to develop a visualisation framework to reveal critical evolutionary patterns of GP [153]. The results show that the proposed algorithm can successfully capture important patterns of GP and show how the algorithm actually works.

To improve the interpretability of the evolutionary process of GP in JSS, two main factors, an effective evolutionary process capture strategy and a fast visualisation tool, need to be

considered. How to capture characteristics of the evolutionary process in tree-based search space of GP is fundamental. An efficient tool to show visualisation is necessary, especially with interactions with human beings.

B. Search Mechanisms

The basic idea of GP is to improve the quality of GP individuals generation by generation. Thus, a good offspring generation scheme is important for GP with machine learning techniques such as the offspring generation with multiple tasks in multitask learning for JSS without being restricted to any predefined structure. On the one hand, the flexible representation of GP makes itself a good candidate for learning scheduling heuristics automatically in JSS. On the other hand, the search of GP is much more challenging as the search space and fitness landscape are much bigger and more complex.

1) Genotype vs Phenotype: Genotype of GP in JSS refers to the structures (explicitly) of GP individuals, while phenotype is mainly about the behaviour (implicitly) of GP individuals in particular problems [250]–[252]. Variation occurs at the genotypic level with GP trees, however, fitness is evaluated at the phenotypic level. A small change in a GP individual can lead to a big change in the behaviour of the GP individual, this makes the search quite challenging. To handle this issue, the mapping between genotype and phenotype is worth investigating to know how variations are effectively translated into quality improvements. The mapping between genotype and phenotype is still an open question. To the best of our knowledge, there is no work on the study of the mapping between genotype and phenotype in JSS.

2) Local Search: Local search algorithms move one solution to another, i.e., neighbours, in the solution space by applying local changes. However, it is challenging to apply local search techniques for GP in JSS since a small change in a GP tree can lead to a big change of the behaviour of a GP individual. In other words, it is non-trivial to define the neighbours of a candidate solution. The mutation operator was utilised as a local search technique for GP in JSS in [217], [253]–[255] based on the assumption that neighbours are derived from the candidate solutions with small changes, i.e., the changes are considered in a genotype level. If the newly generated GP individuals are better than the candidate solutions, the newly generated individuals will replace the candidate solution. However, the changes based on a phenotypic level have not been considered yet. More research on local search with GP is worth investigating.

C. Scalability and Computational Cost

Efficiency is one of the main performance measures for an algorithm. One of the key factors that affect the computational cost of GP in JSS is the scale of the JSS problems. There is a higher computational cost as the scales of problems increase.

Although there are some studies on building surrogate models to reduce the computational cost of GP for JSS, most of these algorithms are used to estimate the individuals in the intermediate population rather than the individuals in the main population directly. Only using surrogate models for fitness estimation in intermediate population aims to improve the performance of an algorithm without increasing the training time. This is different from reducing the training time but maintaining or even improving the performance of an algorithm. The latter way is a more direct approach to reduce the computational cost. However, the latter one is more challenging since it has more effect on the evolutionary process of GP on JSS [173].

Another possible way to reduce the computational cost is to train on small scale problems and apply the learned scheduling to problems with large scale problems. However, based on our preliminary work, the learned scheduling heuristics on small scale problems have unsatisfied adaption ability on large scale problems. Therefore, to reduce the computational cost, designing effective scheduling heuristics adaptation strategies [256] is also an important future task.

D. Interactions in Decision Making

As mentioned earlier, we can use GP to learn ensemble of scheduling heuristics for prioritising machines or operations. However, how to use the ensemble of scheduling heuristics is non-trivial. Existing studies mainly use simple voting or linear combination strategies for using ensemble of rules. The interactions of ensembles have not been fully explored with GP for JSS. If we can capture the characteristics of ensembles well, and gives proper weights of ensemble based on the decision situations by considering the interactions between ensembles, better schedules can be achieved.

Another interaction in decision making is the interaction between machines. The current scheduling methods normally ignore the interactions between machines. In other words, once an operation is allocated to a machine, it will wait for that machine to process it. However, as time goes by, the state of the JSS can be changed, and the moving of operations between machines can be beneficial. A global pool was considered to delay the allocation of operations to machines in [257]. The corresponding GP algorithm achieves better performance than the compared algorithms. However, the computational time of the algorithm is significantly increased, since we need to calculate a large number of priority values for operations in the global pool frequently. How to make effective schedules efficiently by considering the interactions among machines is worth studying.

E. Transferred Knowledge

We have seen that knowledge transfer has been successfully used in GP for JSS with different techniques such as feature selection by transferring knowledge from different evolutionary stages [160], [162], multi-fidelity surrogate by sharing knowledge between different surrogates [174], and multitask learning by sharing knowledge between tasks [168].

An issue is that although the algorithms with knowledge transfer can achieve better performance, it is still not clear what knowledge is exactly transferred, and benefit the performance improvement of the algorithms. Therefore, to study on the improvement mechanism from the perspective of transferred knowledge is an important talk. However, this

is a challenging task. First, how to represent the knowledge for tree-based structures of GP in JSS? Second, how to measure and define the knowledge? In other words, we need to know what knowledge we have. Last but not least, how the transferred knowledge can be used well for different target tasks? In other words, how can the transferred knowledge can be incorporated well with the target tasks.

F. Consistency Between Scheduling Heuristics and Solutions

As we mentioned earlier, GP has used a hyper-heuristic approach to learning scheduling heuristics in JSS. One challenge of using GP as a hyper-heuristic approach is that the consistency between heuristics to solutions is not always consistent [207]. This is especially true in multi-objective learning, since a heuristic aims to handle more than one objective [106]. In other words, when testing a Pareto front obtained from the training process on unseen instances, the corresponding solutions may no longer be exactly non-dominated by each other. One way to handle this issue is to take the consistency of heuristics and solutions into consideration when learning scheduling heuristics. We expect that the learned scheduling heuristics have a good consistency to solutions, and thus improve the generalisation ability of learned scheduling heuristics. However, it is challenging to measure the consistency between scheduling heuristics to solutions, and how it can be considered in the training process.

G. Other Dynamic Events

From literature, job arrivals over time and machine breakdown are two widely studied disturbances for dynamic events in JSS. In practice, there are more disturbances such as changes in job priorities, rush orders, and shortage of materials in JSS [258]. However, they have not been widely investigated yet. A potential research direction is to expand the scheduling problems to consider different dynamic events in JSS. Another interesting research question is to investigate which dynamic event has more impact on making schedules. If we can obtain this information, with multiple dynamic events, we can prioritise jobs to make better schedules.

VIII. Conclusions

This paper provides a comprehensive survey on recent studies of design issues of GP on different types of JSS, and recent machine learning techniques for GP in JSS. This survey shows how GP has been used on different types of JSS problems, including static JSS, dynamic JSS, and dynamic flexible JSS with multiple decision making processes. Detailed design issues such as representation and genetic operators, are provided for different types of JSS. In addition, the recent widely studied machine learning techniques in JSS with GP such as advanced genetic operators, surrogate, feature selection, multitask learning, ensemble learning and multi-objective learning, have been discussed by investigating their advantages and disadvantages, and existing and potential collaborations between them. The values of the studies of GP approaches in JSS have been shown in different applications such as semi-conductor manufacturing. More importantly, issues and challenges of the reviewed studies are discussed to provide more insights into this field, and look ahead to future research.

The contributions of this survey have four folds. First, this survey provides comprehensive discussions of GP in different types of JSS. This makes it easier for researchers to get a good understanding of the field, especially for the beginner. Second, this survey covers the popular ongoing techniques for GP in JSS in recent years, which broadens the research directions for related researchers. This can also guide researchers to incorporate more advanced machine learning techniques for JSS. Third, this survey connects GP with machine learning for scheduling which has a large number of real-world applications. This survey can also benefit the development of ideas for handling other combinatorial optimisation problems. Last, this survey detects a number of promising research directions and possible real-world applications, which can promote the development of the research on GP in JSS.

Although a number of machine learning techniques have achieved success for GP in JSS, there are still challenges and their potential has not been fully investigated. In terms of the search mechanism, it is still not fully understandable of the mapping between the heuristics and solutions, e.g., scheduling heuristics and the generated schedules. This makes it hard to guide the search direction and identify the neighbours of solutions for local search. Another challenging issue is the interpretability of the learned scheduling heuristics and the evolutionary process of GP. Proper measurements with theory analyses of the interpretability of scheduling heuristic design need to be investigated. Regrading multitask learning, the current multitask GP approaches focus on when and how to share knowledge. However, what exactly knowledge is shared has not been answered. Efficiency is also one of the main issues that need to be considered, especially for large scale problems. In addition, better interaction between machines can potentially improve scheduling effectiveness.

References

[1] F. Zhang, S. Nguyen, Y. Mei, and M. Zhang, "Genetic programming for production scheduling: An evolutionary learning approach," in *Machine Learning: Foundations, Methodologies, and Applications*. Springer, 2021, DOI: 10.1007/978-981-16-4859-5, pp. XXXIII+338 pages.

[2] J. Branke, S. Nguyen, C. W. Pickardt, and M. Zhang, "Automated design of production scheduling heuristics: A review," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 1, pp. 110–124, 2015.

[3] S. Kumar, "A study of the supermarket industry and its growing logistics capabilities," *International Journal of Retail & Distribution Management*, 2008.

[4] D. A. Mollenkopf, L. K. Ozanne, and H. J. Stolze, "A transformative supply chain response to covid-19," *Journal of Service Management*, 2020.

[5] A. M. Fathollahi-Fard, A. Ahmadi, F. Goodarzian, and N. Cheikhrouhou, "A bi-objective home healthcare routing and scheduling problem considering patients' satisfaction in a fuzzy environment," *Applied soft computing*, vol. 93, pp. 1–16, 2020.

[6] A. M. Fathollahi-Fard, A. Ahmadi, and B. Karimi, "Sustainable and robust home healthcare logistics: A response to the covid-19 pandemic," *Symmetry*, vol. 14, no. 2, p. 193, 2022.

[7] Q. Zhang, M. Cao, F. Zhang, J. Liu, and X. Li, "Effects of corporate social responsibility on customer satisfaction and organizational attractiveness: A signaling perspective," *Business Ethics: A European Review*, vol. 29, no. 1, pp. 20–34, 2020.

[8] B. Vahedi-Nouri, R. Tavakkoli-Moghaddam, Z. Hanzálek, and A. Dolgui, "Workforce planning and production scheduling in a reconfigurable manufacturing system facing the covid-19 pandemic," *Journal of Manufacturing Systems*, vol. 63, pp. 563–574, 2022.

[9] M. R. Bazargan-Lari, S. Taghipour, A. Zaretalab, and M. Sharifi, "Production scheduling optimization for a parallel machine subject to physical distancing due to covid-19," *Operations Management Research*, pp. 1–25, 2022.

[10] F. Zhang, Y. Mei, S. Nguyen, K. C. Tan, and M. Zhang, "Multitask genetic programming based generative hyper-heuristics: A case study in dynamic scheduling," *IEEE Transactions on Cybernetics*, 2020, Doi: 10.1109/TCYB.2021.3065340.

[11] S. Emde and M. Gendreau, "Scheduling in-house transport vehicles to feed parts to automotive assembly lines," *European Journal of Operational Research*, vol. 260, no. 1, pp. 255–267, 2017.

[12] C. L. Wu and S. X. Lim, "Effects of enterprise bargaining and agreement clauses on the operating cost of airline ground crew scheduling," *Journal of Air Transport Management*, vol. 91, p. 101972, 2021.

[13] H. Abouee Mehrizi, A. Aminoleslami, J. Darko, E. Osei, and H. Mahmoudzadeh, "Staff scheduling during a pandemic: The case of radiation therapy department," *Social Science Research Network*, 2022.

[14] J. A. Gromicho, J. J. Van Hoorn, F. Saldanha-da Gama, and G. T. Timmer, "Solving the job-shop scheduling problem optimally by dynamic programming," *Computers & Operations Research*, vol. 39, no. 12, pp. 2968–2977, 2012.

[15] D. Ouelhadj and S. Petrovic, "A survey of dynamic scheduling in manufacturing systems," *Journal of Scheduling*, vol. 12, no. 4, pp. 417–431, 2009.

[16] I. A. Chaudhry and A. A. Khan, "A research survey: review of flexible job shop scheduling techniques," *International Transactions in Operational Research*, vol. 23, no. 3, pp. 551–591, 2016.

[17] A. Türkyılmaz, Ö. Şenvar, İ. Ünal, and S. Bulkan, "A research survey: heuristic approaches for solving multi objective flexible job shop problems," *Journal of Intelligent Manufacturing*, vol. 31, pp. 1949–1983, 2020.

[18] M. Durasevic and D. Jakobovic, "A survey of dispatching rules for the dynamic unrelated machines environment," *Expert Systems with Applications*, vol. 113, pp. 555–569, 2018.

[19] J. H. Blackstone, D. T. Phillips, and G. L. Hogg, "A state-of-the-art survey of dispatching rules for manufacturing job shop operations," *The International Journal of Production Research*, vol. 20, no. 1, pp. 27–45, 1982.

[20] S. Nguyen, Y. Mei, B. Xue, and M. Zhang, "A hybrid genetic programming algorithm for automated design of dispatching rules," *Evolutionary Computation*, pp. 1–31, 2018.

[21] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Genetic programming for job shop scheduling," in *Evolutionary and Swarm Intelligence Algorithms*. Springer, 2019, pp. 143–167.

[22] S. Nguyen, D. Thiruvady, A. Ernst, and D. Alahakoon, "Genetic programming approach to learning multi-pass heuristics for resource constrained job scheduling," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018, pp. 1167–1174.

[23] F. Zhang, "Genetic programming hyper-heuristics for dynamic flexible job shop scheduling," *Victoria University of Wellington*, 2021. Doi: https://doi.org/10.26686/wgtn.16528677.

[24] S. Nguyen, D. Thiruvady, M. Zhang, and D. Alahakoon, "Automated design of multipass heuristics for resource-constrained job scheduling with self-competitive genetic programming," *IEEE Transactions on Cybernetics*, 2021, DOI: 10.1109/TCYB.2021.3062799.

[25] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Correlation coefficient based recombinative guidance for genetic programming hyper-heuristics in dynamic flexible job shop scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 3, pp. 552–566, 2021.

[26] L. Planinic, M. DJurasevic, and D. Jakobovic, "On the application of ϵ-lexicase selection in the generation of dispatching rules," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2021, pp. 2125–2132.

[27] M. DJurasevic, L. Planinic, F. J. G. Gala, and D. Jakobovic, "Novel ensemble collaboration method for dynamic scheduling problems," *Proceedings of the Genetic and Evolutionary Computation Conference*, 2022.

[28] Y. Zeitrag, J. R. Figueira, N. Horta, and R. Neves, "Surrogate-assisted automatic evolving of dispatching rules for multi-objective dynamic job shop scheduling using genetic programming," *Expert Systems with Applications*, p. 118194, 2022.

[29] S. Shady, T. Kaihara, N. Fujii, and D. Kokuryo, "A novel feature selection for evolving compact dispatching rules using genetic programming for dynamic job shop scheduling," *International Journal of Production Research*, pp. 1–24, 2022.

[30] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 2, pp. 193–208, 2014.

[31] S. Nguyen, Y. Mei, and M. Zhang, "Genetic programming for production scheduling: a survey with a unified framework," *Complex & Intelligent Systems*, vol. 3, no. 1, pp. 41–66, 2017.

[32] X. Wu, L. Xiao, Y. Sun, J. Zhang, T. Ma, and L. He, "A survey of human-in-the-loop for machine learning," *Future Generation Computer Systems*, 2022.

[33] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: a methodological tour d'horizon," *European Journal of Operational Research*, vol. 290, no. 2, pp. 405–421, 2021.

[34] L. Nie, L. Gao, P. Li, and X. Li, "A gep-based reactive scheduling policies constructing approach for dynamic flexible job shop scheduling problem with job release dates," *Journal of Intelligent Manufacturing*, vol. 24, no. 4, pp. 763–774, 2013.

[35] X. N. Shen and X. Yao, "Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems," *Information Sciences*, vol. 298, pp. 198–224, 2015.

[36] X. Cai and S. Zhou, "Stochastic scheduling on parallel machines subject to random breakdowns to minimize expected costs for earliness and tardy jobs," *Operations Research*, vol. 47, no. 3, pp. 422–437, 1999.

[37] T. Hildebrandt and J. Branke, "On using surrogates with genetic programming," *Evolutionary Computation*, vol. 23, no. 3, pp. 343–367, 2015.

[38] F. Zhang, Y. Mei, and M. Zhang, "Can stochastic dispatching rules evolved by genetic programming hyper-heuristics help in dynamic flexible job shop scheduling?" in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2019, pp. 41–48.

[39] Z. Wang, J. Zhang, and S. Yang, "An improved particle swarm optimization algorithm for dynamic job shop scheduling problems with random job arrivals," *Swarm and Evolutionary Computation*, vol. 51, p. 100594, 2019.

[40] J. Chang, D. Yu, Y. Hu, W. He, and H. Yu, "Deep reinforcement learning for dynamic flexible job shop scheduling with random job arrival," *Processes*, vol. 10, no. 4, p. 760, 2022.

[41] J. Park, Y. Mei, S. Nguyen, G. Chen, and M. Zhang, "Investigating a machine breakdown genetic programming approach for dynamic job shop scheduling," in *Proceedings of the European Conference on Genetic Programming*. Springer, 2018, pp. 253–270.

[42] G. Zhang, X. Lu, X. Liu, L. Zhang, S. Wei, and W. Zhang, "An effective two-stage algorithm based on convolutional neural network for the bi-objective flexible job shop scheduling problem with machine breakdown," *Expert Systems with Applications*, vol. 203, p. 117460, 2022.

[43] J. Park, Y. Mei, S. Nguyen, G. Chen, and M. Zhang, "Investigating the generality of genetic programming based hyper-heuristic approach to dynamic job shop scheduling with machine breakdown," in *Australasian Conference on Artificial Life and Computational Intelligence*. Springer, 2017, pp. 301–313.

[44] J. Xie, L. Gao, K. Peng, X. Li, and H. Li, "Review on flexible job shop scheduling," *IET Collaborative Intelligent Manufacturing*, vol. 1, no. 3, pp. 67–77, 2019.

[45] K. Gao, Z. Cao, L. Zhang, Z. Chen, Y. Han, and Q. Pan, "A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 4, pp. 904–916, 2019.

[46] S. Jun, S. Lee, and H. Chun, "Learning dispatching rules using random forest in flexible job shop scheduling problems," *International Journal of Production Research*, vol. 57, no. 10, pp. 3290–3310, 2019.

[47] S. M. Lee and A. A. Asllani, "Job scheduling with dual criteria and sequence-dependent setups: mathematical versus genetic programming," *Omega*, vol. 32, no. 2, pp. 145–153, 2004.

[48] P. Sharma and A. Jain, "A review on job shop scheduling with setup times," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 230, no. 3, pp. 517–533, 2016.

[49] A. S. Jain and S. Meeran, "Deterministic job-shop scheduling: Past, present and future," *European journal of operational research*, vol. 113, no. 2, pp. 390–434, 1999.

[50] P. Brucker, B. Jurisch, and B. Sievers, "A branch and bound algorithm for the job-shop scheduling problem," *Discrete applied mathematics*, vol. 49, no. 1-3, pp. 107–127, 1994.

[51] S. Ashour and S. Hiremath, "A branch-and-bound approach to the job-shop scheduling problem," *International Journal of Production Research*, vol. 11, no. 1, pp. 47–58, 1973.

[52] H. Chen, C. Chu, and J. Proth, "An improvement of the lagrangean relaxation approach for job shop scheduling: a dynamic programming method," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 5, pp. 786–795, 1998.

[53] A. Jamili, "Robust job shop scheduling problem: Mathematical models, exact and heuristic algorithms," *Expert systems with Applications*, vol. 55, pp. 341–350, 2016.

[54] M. Dell'Amico and M. Trubian, "Applying tabu search to the job-shop scheduling problem," *Annals of Operations research*, vol. 41, no. 3, pp. 231–252, 1993.

[55] F. Pezzella, G. Morganti, and G. Ciaschetti, "A genetic algorithm for the flexible job-shop scheduling problem," *Computers & Operations Research*, vol. 35, no. 10, pp. 3202–3212, 2008.

[56] M. K. Amjad, S. I. Butt, R. Kousar, R. Ahmad, M. H. Agha, Z. Faping, N. Anjum, and U. Asgher, "Recent research trends in genetic algorithm based flexible job shop scheduling problems," *Mathematical Problems in Engineering*, vol. 2018, pp. 1–32, 2018.

[57] D. Sha and H.-H. Lin, "A multi-objective pso for job-shop scheduling problems," *Expert Systems with Applications*, vol. 37, no. 2, pp. 1065–1070, 2010.

[58] T. Jamrus, C. F. Chien, M. Gen, and K. Sethanan, "Hybrid particle swarm optimization combined with genetic operators for flexible job-shop scheduling under uncertain processing time for semiconductor manufacturing," *IEEE Transactions on Semiconductor Manufacturing*, vol. 31, no. 1, pp. 32–41, 2017.

[59] B. Çaliş and S. Bulkan, "A research survey: review of ai solution strategies of job shop scheduling problem," *Journal of Intelligent Manufacturing*, vol. 26, no. 5, pp. 961–973, 2015.

[60] P. D. Dominic, S. Kaliyamoorthy, and M. S. Kumar, "Efficient dispatching rules for dynamic job shop scheduling," *The International Journal of Advanced Manufacturing Technology*, vol. 24, no. 1-2, pp. 70–75, 2004.

[61] A. Kaban, Z. Othman, and D. Rohmah, "Comparison of dispatching rules in job-shop scheduling problem using simulation: a case study," *International Journal of Simulation Modelling*, vol. 11, no. 3, pp. 129–140, 2012.

[62] M. Habib Zahmani and B. Atmani, "Multiple dispatching rules allocation in real time using data mining, genetic algorithms, and simulation," *Journal of Scheduling*, vol. 24, no. 2, pp. 175–196, 2021.

[63] F. Zhang, Y. Mei, and M. Zhang, "A new representation in genetic programming for evolving dispatching rules for dynamic flexible job shop scheduling," in *Proceedings of the European Conference on Evolutionary Computation in Combinatorial Optimization*. Springer, 2019, pp. 33–49.

[64] N. Pillay and R. Qu, *Hyper-heuristics: Theory and applications*. Springer, 2018.

[65] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, "Hyper-heuristics: A survey of the state of the art," *Journal of the Operational Research Society*, vol. 64, no. 12, pp. 1695–1724, 2013.

[66] E. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, and S. Schulenburg, "Hyper-heuristics: An emerging direction in modern search technology," in *Handbook of metaheuristics*. Springer, 2003, pp. 457–474.

[67] S. Shady, T. Kaihara, N. Fujii, and D. Kokuryo, "A hyper-heuristic framework using gp for dynamic job shop scheduling problem," in *Proceedings of the Annual Conference of the Institute of Systems, Control and Information Engineers*, 2020, pp. 248–252.

[68] S. Nguyen, Y. Mei, H. Ma, A. Chen, and M. Zhang, "Evolutionary scheduling and combinatorial optimisation: Applications, challenges, and future directions," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2016, pp. 3053–3060.

[69] J. H. Drake, A. Kheiri, E. Özcan, and E. K. Burke, "Recent advances in selection hyper-heuristics," *European Journal of Operational Research*, vol. 285, no. 2, pp. 405–428, 2020.

[70] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward, "A classification of hyper-heuristic approaches: revisited," in *Handbook of metaheuristics*. Springer, 2019, pp. 453–477.

[71] B. Cunha, A. M. Madureira, B. Fonseca, and D. Coelho, "Deep reinforcement learning as a job shop scheduling solver: A literature review," in *Proceedings of the International Conference on Hybrid Intelligent Systems*. Springer, 2020, pp. 350–359.

[72] R. Liu, R. Piplani, and C. Toro, "Deep reinforcement learning for dynamic scheduling of a flexible job shop," *International Journal of Production Research*, vol. 60, no. 13, pp. 4049–4069, 2022.

[73] S. Luo, L. Zhang, and Y. Fan, "Dynamic multi-objective scheduling for flexible job shop by deep reinforcement learning," *Computers & Industrial Engineering*, vol. 159, p. 107489, 2021.

[74] B. Han and J. Yang, "A deep reinforcement learning based solution for flexible job shop scheduling problem," *International Journal of Simulation Modelling*, vol. 20, no. 2, pp. 375–386, 2021.

[75] Z. H. Zhou, *Machine learning*. Springer Nature, 2021.

[76] G. Bonaccorso, *Machine learning algorithms*. Packt Publishing Ltd, 2017.

[77] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job shop scheduling," *IEEE Transactions on Cybernetics*, vol. 51, no. 4, pp. 1797–1811, 2021.

[78] K. Miyashita, "Job-shop scheduling with genetic programming," in *Proceedings of the Annual Conference on Genetic and Evolutionary Computation*. Morgan Kaufmann Publishers Inc., 2000, pp. 505–512.

[79] S. Shady, T. Kaihara, N. Fujii, and D. Kokuryo, "Automatic design of dispatching rules with genetic programming for dynamic job shop scheduling," in *IFIP International Conference on Advances in Production Management Systems*. Springer, 2020, pp. 399–407.

[80] R. Braune, F. Benda, K. F. Doerner, and R. F. Hartl, "A genetic programming learning approach to generate dispatching rules for flexible shop scheduling problems," *International Journal of Production Economics*, vol. 243, p. 108342, 2022.

[81] T. Bäck, D. B. Fogel, and Z. Michalewicz, *Evolutionary computation 1: Basic algorithms and operators*. CRC press, 2018.

[82] S. Nguyen and D. Thiruvady, "Evolving large reusable multi-pass heuristics for resource constrained job scheduling," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2020, pp. 1–8.

[83] Y. Mei, S. Nguyen, and M. Zhang, "Evolving time-invariant dispatching rules in job shop scheduling with genetic programming," in *Proceedings of the European Conference on Genetic Programming*. Springer, 2017, pp. 147–163.

[84] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Genetic programming with adaptive search based on the frequency of features for dynamic flexible job shop scheduling," in *Proceedings of the European Conference on Evolutionary Computation in Combinatorial Optimization*. Springer, 2020, pp. 214–230.

[85] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 5, pp. 621–639, 2013.

[86] Z. Huang, F. Zhang, Y. Mei, and M. Zhang, "An investigation of multi-task linear genetic programming for dynamic job shop scheduling," in *n Proceedings of the Genetic and Evolutionary Computation Conference*. Springer, 2022, pp. 162–178.

[87] L. Nie, Y. Bai, X. Wang, and K. Liu, "Discover scheduling strategies with gene expression programming for dynamic flexible job shop scheduling problem," in *Proceedings of the International Conference in Swarm Intelligence*. Springer, 2012, pp. 383–390.

[88] G. Ozturk, O. Bahadir, and A. Teymourifar, "Extracting priority rules for dynamic multi-objective flexible job shop scheduling problems using gene expression programming," *International Journal of Production Research*, vol. 57, no. 10, pp. 3121–3137, 2019.

[89] L. Nie, L. Gao, P. Li, and L. Zhang, "Application of gene expression programming on dynamic job shop scheduling problem," in *Proceedings of the Conference on Computer Supported Cooperative Work in Design*. IEEE, 2011, pp. 291–295.

[90] J. Zhong, L. Feng, and Y. S. Ong, "Gene expression programming: A survey," *IEEE Computational Intelligence Magazine*, vol. 12, no. 3, pp. 54–72, 2017.

[91] E. Taillard, "Benchmarks for basic scheduling problems," *European Journal of Operational Research*, vol. 64, no. 2, pp. 278–285, 1993.

[92] M. M. Gohareh and E. Mansouri, "A simulation-optimization framework for generating dynamic dispatching rules for stochastic job shop with earliness and tardiness penalties," *Computers & Operations Research*, vol. 140, p. 105650, 2022.

[93] L. Cai, W. Li, Y. Luo, and L. He, "Real-time scheduling simulation optimisation of job shop in a production-logistics collaborative environment," *International Journal of Production Research*, pp. 1–21, 2022.

[94] Y. F. Wang, "Adaptive job shop scheduling strategy based on weighted q-learning algorithm," *Journal of Intelligent Manufacturing*, vol. 31, no. 2, pp. 417–432, 2020.

[95] H. Fan, H. Xiong, and M. Goh, "Genetic programming-based hyper-heuristic approach for solving dynamic job shop scheduling problem

with extended technical precedence constraints," *Computers & Operations Research*, vol. 134, p. 105401, 2021.

[96] J. R. Koza, "Genetic programming as a means for programming computers by natural selection," *Statistics and Computing*, vol. 4, no. 2, pp. 87–112, 1994.

[97] T. Helmuth and A. Abdelhady, "Benchmarking parent selection for program synthesis by genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2020, pp. 237–238.

[98] H. Xie and M. Zhang, "Parent selection pressure auto-tuning for tournament selection in genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 1, pp. 1–19, 2012.

[99] Y. Fang and J. Li, "A review of tournament selection in genetic programming," in *International symposium on intelligence computation and applications*. Springer, 2010, pp. 181–192.

[100] G. Shi, F. Zhang, and Y. Mei, "A novel fitness function for genetic programming in dynamic flexible job shop scheduling," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2022. DOI: 10.1109/CEC55065.2022.9870235.

[101] M. Xu, Y. Mei, F. Zhang, and Z. Mengjie, "Genetic programming with cluster selection for dynamic flexible job shop scheduling," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2022. DOI: 10.1109/CEC55065.2022.9870431.

[102] ——, "Genetic programming with diverse partner selection for dynamic flexible job shop scheduling," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2022. DOI: 10.1145/3520304.3528920.

[103] ——, "Genetic programming with multi-case fitness for dynamic flexible job shop scheduling," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2022. DOI: 10.1109/CEC55065.2022.9870340.

[104] M. Xu, F. Zhang, Y. Mei, and M. Zhang, "Genetic programming with archive for dynamic flexible job shop scheduling," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2021, pp. 2117–3109.

[105] K. E. Kinnear, W. B. Langdon, L. Spector, P. J. Angeline, and U.-M. O'Reilly, *Advances in genetic programming*. MIT press, 1994, vol. 3.

[106] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Multitask multiobjective genetic programming for automated scheduling heuristic learning in dynamic flexible job-shop scheduling," *IEEE Transactions on Cybernetics*, 2022. DOI: 10.1109/TCYB.2022.3196887.

[107] C. Ferreira, G. Figueira, and P. Amorim, "Optimizing dispatching rules for stochastic job shop scheduling," in *International Conference on Hybrid Intelligent Systems*. Springer, 2018, pp. 321–330.

[108] H. Assimi, A. Jamali, and N. Nariman-Zadeh, "Multi-objective sizing and topology optimization of truss structures using genetic programming based on a new adaptive mutant operator," *Neural Computing and Applications*, vol. 31, no. 10, pp. 5729–5749, 2019.

[109] M. H. Kim, R. I. B. McKay, N. X. Hoai, and K. Kim, "Operator self-adaptation in genetic programming," in *Proceedings of the European Conference on Genetic Programming*. Springer, 2011, pp. 215–226.

[110] H. Xie, M. Zhang, and P. Andreae, "An analysis of depth of crossover points in tree-based genetic programming," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2007, pp. 4561–4568.

[111] U. M. O'Reilly and F. Oppacher, "Program search with a hierarchical variable length representation: Genetic programming, simulated annealing and hill climbing," in *International Conference on Parallel Problem Solving from Nature*. Springer, 1994, pp. 397–406.

[112] T. Ito, H. Iba, and S. Sato, "A self-tuning mechanism for depth-dependent crossover," *Advances in Genetic Programming*, vol. 3, p. 377, 1999.

[113] Q. U. Nguyen, T. A. Pham, X. H. Nguyen, and J. McDermott, "Subtree semantic geometric crossover for genetic programming," *Genetic Programming and Evolvable Machines*, vol. 17, no. 1, pp. 25–53, 2016.

[114] Y. Mei, S. Nguyen, and M. Zhang, "Constrained dimensionally aware genetic programming for evolving interpretable dispatching rules in dynamic job shop scheduling," in *Proceedings of the Asia-Pacific Conference on Simulated Evolution and Learning*. Springer, 2017, pp. 435–447.

[115] N. F. McPhee, M. K. Dramdahl, and D. Donatucci, "Impact of crossover bias in genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2015, pp. 1079–1086.

[116] R. Poli and N. F. McPhee, "General schema theory for genetic programming with subtree-swapping crossover: Part I," *Evolutionary Computation*, vol. 11, no. 1, pp. 53–66, 2003.

[117] ——, "General schema theory for genetic programming with subtree-swapping crossover: Part II," *Evolutionary Computation*, vol. 11, no. 2, pp. 169–206, 2003.

[118] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Guided subtree selection for genetic operators in genetic programming for dynamic flexible job shop scheduling," in *Proceedings of the European Conference on Genetic Programming*. Springer, 2020, pp. 262–278.

[119] K. Jaklinovic, M. Durasevic, and D. Jakobovic, "Designing dispatching rules with genetic programming for the unrelated machines environment with constraints," *Expert Systems with Applications*, p. 114548, 2020.

[120] D. Yska, Y. Mei, and M. Zhang, "Genetic programming hyper-heuristic with cooperative coevolution for dynamic flexible job shop scheduling," in *European Conference on Genetic Programming*. Springer, 2018, pp. 306–321.

[121] N. Kundakcı and O. Kulak, "Hybrid genetic algorithms for minimizing makespan in dynamic job shop scheduling problem," *Computers & Industrial Engineering*, vol. 96, pp. 31–51, 2016.

[122] H. Zhang, B. Buchmeister, X. Li, and R. Ojstersek, "Advanced meta-heuristic method for decision-making in a dynamic job shop scheduling environment," *Mathematics*, vol. 9, no. 8, p. 909, 2021.

[123] S. Panda and Y. Mei, "Genetic programming with algebraic simplification for dynamic job shop scheduling," in *in Proceedings of Congress on Evolutionary Computation*. IEEE, 2021, pp. 1848–1855.

[124] J. Zhang, G. Ding, Y. Zou, S. Qin, and J. Fu, "Review of job shop scheduling research and its new perspectives under industry 4.0," *Journal of Intelligent Manufacturing*, vol. 30, no. 4, pp. 1809–1830, 2019.

[125] S. Panda, Y. Mei, and M. Zhang, "Simplifying dispatching rules in genetic programming for dynamic job shop scheduling," in *in Proceedings of European Conference on Evolutionary Computation in Combinatorial Optimization*. Springer, 2022, pp. 95–110.

[126] C. Dimopoulos and A. Zalzala, "A genetic programming heuristic for the one-machine total tardiness problem," in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, vol. 3. IEEE, 1999, pp. 2207–2214.

[127] W.-J. Yin, M. Liu, and C. Wu, "Learning single-machine scheduling heuristics subject to machine breakdowns with genetic programming," in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 2. IEEE, 2003, pp. 1050–1055.

[128] D. Jakobović and L. Budin, "Dynamic scheduling with genetic programming," in *Proceedings of the European Conference on Genetic Programming*. Springer, 2006, pp. 73–84.

[129] D. Jakobović and K. Marasović, "Evolving priority scheduling heuristics with genetic programming," *Applied Soft Computing*, vol. 12, no. 9, pp. 2781–2789, 2012.

[130] C. Dimopoulos and A. M. Zalzala, "Investigating the use of genetic programming for a classic one-machine scheduling problem," *Advances in Engineering Software*, vol. 32, no. 6, pp. 489–498, 2001.

[131] F. J. Gil-Gala, M. R. Sierra, C. Mencía, and R. Varela, "Genetic programming with local search to evolve priority rules for scheduling jobs on a machine with time-varying capacity," *Swarm and Evolutionary Computation*, vol. 66, p. 100944, 2021.

[132] S. Nguyen, "Optimization, dispatching rules and hyper-heuristics: A comparison in dynamic single machine scheduling," in *Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2017, pp. 4796–4801.

[133] S. Grabowska, "Smart factories in the age of industry 4.0," *Management systems in production engineering*, 2020.

[134] D. Mourtzis, J. Angelopoulos, and G. Dimitrakopoulos, "Design and development of a flexible manufacturing cell in the concept of learning factory paradigm for the education of generation 4.0 engineers," *Procedia Manufacturing*, vol. 45, pp. 361–366, 2020.

[135] T. Hildebrandt, J. Heger, and B. Scholz-Reiter, "Towards improved dispatching rules for complex shop floor scenarios: a genetic programming approach," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2010, pp. 257–264.

[136] D. Jakobović, L. Jelenković, and L. Budin, "Genetic programming heuristics for multiple machine scheduling," in *Proceedings of the European Conference on Genetic Programming*. Springer, 2007, pp. 321–330.

[137] M. Durasevic, D. Jakobovic, and K. Knezevic, "Adaptive scheduling on unrelated machines with genetic programming," *Applied Soft Computing*, vol. 48, pp. 419–430, 2016.

[138] N. Ishii and J. J. Talavage, "A mixed dispatching rule approach in fms scheduling," *International Journal of Flexible Manufacturing Systems*, vol. 6, no. 1, pp. 69–87, 1994.

[139] N. Kim, S. Barde, K. Bae, and H. Shin, "Learning per-machine linear dispatching rule for heterogeneous multi-machines control," *International Journal of Production Research*, pp. 1–21, 2021.

[140] B. Mihoubi, B. Bouzouia, and M. Gaham, "Reactive scheduling approach for solving a realistic flexible job shop scheduling problem," *International Journal of Production Research*, vol. 59, no. 19, pp. 5790–5808, 2021.

[141] Y. Li, W. Gu, M. Yuan, and Y. Tang, "Real-time data-driven dynamic scheduling for flexible job shop with insufficient transportation resources using hybrid deep q network," *Robotics and Computer-Integrated Manufacturing*, vol. 74, p. 102283, 2022.

[142] M. Shahgholi Zadeh, Y. Katebi, and A. Doniavi, "A heuristic model for dynamic flexible job shop scheduling problem considering variable processing times," *International Journal of Production Research*, vol. 57, no. 10, pp. 3020–3035, 2019.

[143] C. M. Joo and B. S. Kim, "Hybrid genetic algorithms with dispatching rules for unrelated parallel machine scheduling with setup time and production availability," *Computers & Industrial Engineering*, vol. 85, pp. 102–109, 2015.

[144] A. Baykasoğlu, F. S. Madenoğlu, and A. Hamzadayı, "Greedy randomized adaptive search for dynamic flexible job-shop scheduling," *Journal of Manufacturing Systems*, vol. 56, pp. 425–451, 2020.

[145] K. Li, Q. Deng, L. Zhang, Q. Fan, G. Gong, and S. Ding, "An effective mcts-based algorithm for minimizing makespan in dynamic flexible job shop scheduling problem," *Computers & Industrial Engineering*, vol. 155, p. 107211, 2021.

[146] F. Geyik and A. T. Dosdoğru, "Process plan and part routing optimization in a dynamic flexible job shop scheduling environment: an optimization via simulation approach," *Neural Computing and Applications*, vol. 23, no. 6, pp. 1631–1641, 2013.

[147] J. C. Tay and N. B. Ho, "Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems," *Computers & Industrial Engineering*, vol. 54, no. 3, pp. 453–473, 2008.

[148] F. Zhang, Y. Mei, and M. Zhang, "Genetic programming with multi-tree representation for dynamic flexible job shop scheduling," in *Proceedings of the Australasian Joint Conference on Artificial Intelligence*. Springer, 2018, pp. 472–484.

[149] Y. Zhou, J. J. Yang, and L. Y. Zheng, "Hyper-heuristic coevolution of machine assignment and job sequencing rules for multi-objective dynamic flexible job shop scheduling," *IEEE Access*, vol. 7, pp. 68–88, 2018.

[150] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Learning strategies on scheduling heuristics of genetic programming in dynamic flexible job shop scheduling," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2022, pp. XXX–XXX.

[151] ——, "Importance-aware genetic programming for automated scheduling heuristics learning in dynamic flexible job shop scheduling," in *Proceedings of the Parallel Problem Solving from Nature*. Springer, 2022, pp. 48–62.

[152] S. Nguyen, M. Zhang, and K. C. Tan, "Adaptive charting genetic programming for dynamic flexible job shop scheduling," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018, pp. 1159–1166.

[153] S. Nguyen, M. Zhang, D. Alahakoon, and K. C. Tan, "Visualizing the evolution of computer programs for genetic programming," *IEEE Computational Intelligence Magazine*, vol. 13, no. 4, pp. 77–94, 2018.

[154] B. Xue, M. Zhang, W. N. Browne, and X. Yao, "A survey on evolutionary computation approaches to feature selection," *IEEE Transactions Evolutionary Computation*, vol. 20, no. 4, pp. 606–626, 2016.

[155] A. Bommert, X. Sun, B. Bischl, J. Rahnenführer, and M. Lang, "Benchmark for filter methods for feature selection in high-dimensional classification data," *Computational Statistics & Data Analysis*, vol. 143, p. 106839, 2020.

[156] S. Shady, T. Kaihara, N. Fujii, and D. Kokuryo, "A new representation and adaptive feature selection for evolving compact dispatching rules for dynamic job shop scheduling with genetic programming," in *IFIP International Conference on Advances in Production Management Systems*. Springer, 2021, pp. 646–654.

[157] ——, "Feature selection approach for evolving reactive scheduling policies for dynamic job shop scheduling problem using gene expression programming," *International Journal of Production Research*, pp. 1–24, 2022.

[158] Y. Mei, M. Zhang, and S. Nyugen, "Feature selection in evolving job shop dispatching rules with genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2016, pp. 365–372.

[159] A. Masood, G. Chen, and M. Zhang, "Feature selection for evolving many-objective job shop scheduling dispatching rules with genetic programming," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2021, pp. 644–651.

[160] Y. Mei, S. Nguyen, B. Xue, and M. Zhang, "An efficient feature selection algorithm for evolving job shop scheduling rules with genetic programming," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 1, no. 5, pp. 339–353, 2017.

[161] Y. Zakaria, Y. Zakaria, A. BahaaElDin, and M. Hadhoud, "Niching-based feature selection with multi-tree genetic programming for dynamic flexible job shop scheduling," in *Proceedings of the International Joint Conference on Computational Intelligence*. Springer, 2019, pp. 3–27.

[162] F. Zhang, Y. Mei, and M. Zhang, "A two-stage genetic programming hyper-heuristic approach with feature selection for dynamic flexible job shop scheduling," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2019, pp. 347–355.

[163] D. Yska, Y. Mei, and M. Zhang, "Feature construction in genetic programming hyper-heuristic for dynamic flexible job shop scheduling," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2018, pp. 149–150.

[164] Y. Jin, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," *Swarm and Evolutionary Computation*, vol. 1, no. 2, pp. 61–70, 2011.

[165] J. González, I. Rojas, J. Ortega, H. Pomares, F. J. Fernandez, and A. F. Díaz, "Multiobjective evolutionary optimization of the size, shape, and position parameters of radial basis function networks for function approximation," *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1478–1495, 2003.

[166] G. Chen, K. Zhang, X. Xue, L. Zhang, C. Yao, J. Wang, and J. Yao, "A radial basis function surrogate model assisted evolutionary algorithm for high-dimensional expensive optimization problems," *Applied Soft Computing*, vol. 116, p. 108353, 2022.

[167] M. Zhao, K. Zhang, G. Chen, X. Zhao, C. Yao, H. Sun, Z. Huang, and J. Yao, "A surrogate-assisted multi-objective evolutionary algorithm with dimension-reduction for production optimization," *Journal of Petroleum Science and Engineering*, vol. 192, p. 107192, 2020.

[168] F. Zhang, Y. Mei, S. Nguyen, M. Zhang, and K. C. Tan, "Surrogate-assisted evolutionary multitask genetic programming for dynamic flexible job shop scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 4, pp. 651–665, 2021.

[169] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Selection schemes in surrogate-assisted genetic programming for job shop scheduling," in *Proceedings of the Asia-Pacific Conference on Simulated Evolution and Learning*. Springer, 2014, pp. 656–667.

[170] F. Zhang, Y. Mei, S. Nguyen, K. C. Tan, and M. Zhang, "Instance rotation based surrogate in genetic programming with brood recombination for dynamic job shop scheduling," *IEEE Transactions on Evolutionary Computation*, 2022, Doi: 10.1109/TEVC.2022.3180693.

[171] S. Nguyen, M. Zhang, and K. C. Tan, "Surrogate-assisted genetic programming with simplified models for automated design of dispatching rules," *IEEE Transactions on Cybernetics*, vol. 47, no. 9, pp. 2951–2965, 2017.

[172] Y. Zhou, J. Yang, and Z. Huang, "Automatic design of scheduling policies for dynamic flexible job shop scheduling via surrogate-assisted cooperative co-evolution genetic programming," *International Journal of Production Research*, vol. 58, no. 9, pp. 2561–2580, 2020.

[173] F. Zhang, Y. Mei, and M. Zhang, "Surrogate-assisted genetic programming for dynamic flexible job shop scheduling," in *Proceedings of the Australasian Joint Conference on Artificial Intelligence*. Springer, 2018, pp. 766–772.

[174] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Collaborative multi-fidelity based surrogate models for genetic programming in dynamic flexible job shop scheduling," *IEEE Transactions on Cybernetics*, 2021. Doi: 10.1109/TCYB.2021.3050141.

[175] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1249, 2018.

[176] J. Park, S. Nguyen, M. Zhang, and M. Johnston, "A single population genetic programming based ensemble learning approach to job shop scheduling," in *Proceedings of the Companion Publication of the Annual Conference on Genetic and Evolutionary Computation*, 2015, pp. 1451–1452.

[177] M. Dumić and D. Jakobović, "Ensembles of priority rules for resource constrained project scheduling problem," *Applied Soft Computing*, p. 107606, 2021.

[178] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 4, pp. 463–484, 2011.

[179] J. Park, S. Nguyen, M. Zhang, and M. Johnston, "Evolving ensembles of dispatching rules using genetic programming for job shop scheduling," in *Proceedings of the European Conference on Genetic Programming*. Springer, 2015, pp. 92–104.

[180] J. Park, Y. Mei, S. Nguyen, G. Chen, M. Johnston, and M. Zhang, "Genetic programming based hyper-heuristics for dynamic job shop scheduling: cooperative coevolutionary approaches," in *Proceedings of the European Conference on Genetic Programming*. Springer, 2016, pp. 115–132.

[181] J. Park, Y. Mei, S. Nguyen, G. Chen, and M. Zhang, "An investigation of ensemble combination schemes for genetic programming based hyper-heuristic approaches to dynamic job shop scheduling," *Applied Soft Computing*, vol. 63, pp. 72–86, 2018.

[182] M. Durasevic and D. Jakobovic, "Comparison of ensemble learning methods for creating ensembles of dispatching rules for the unrelated machines environment," *Genetic Programming and Evolvable Machines*, vol. 19, no. 1, pp. 53–92, 2018.

[183] ——, "Creating dispatching rules by simple ensemble combination," *Journal of Heuristics*, vol. 25, no. 6, pp. 959–1013, 2019.

[184] F. J. Gil-Gala, M. R. Sierra, C. Mencía, and R. Varela, "Combining hyper-heuristics to evolve ensembles of priority rules for on-line scheduling," *Natural Computing*, pp. 1–11, 2020.

[185] M. DJurasevic, L. Planinic, F. J. Gil Gala, and D. Jakobovic, "Constructing ensembles of dispatching rules for multi-objective problems," in *Proceedings of the International Work-Conference on the Interplay Between Natural and Artificial Computation*. Springer, 2022, pp. 119–129.

[186] J. Park, Y. Mei, G. Chen, and M. Zhang, "Niching genetic programming based hyper-heuristic approach to dynamic job shop scheduling: an investigation into distance metrics," in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*. ACM, 2016, pp. 109–110.

[187] E. Hart and K. Sim, "A hyper-heuristic ensemble method for static job-shop scheduling," *Evolutionary Computation*, vol. 24, no. 4, pp. 609–635, 2016.

[188] F. J. Gil Gala, C. Mencía, M. R. Sierra, and R. Varela, "Learning ensembles of priority rules for online scheduling by hybrid evolutionary algorithms," *Integrated Computer-Aided Engineering*, vol. 28, no. 1, pp. 65–80, 2021.

[189] A. Gupta, Y.-S. Ong, and L. Feng, "Multifactorial evolution: toward evolutionary multitasking," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 3, pp. 343–357, 2015.

[190] A. Gupta, Y. Ong, L. Feng, and K. C. Tan, "Multiobjective multifactorial optimization in evolutionary multitasking," *IEEE Transactions on Cybernetics*, vol. 47, no. 7, pp. 1652–1665, 2017.

[191] A. Gupta, Y. S. Ong, and L. Feng, "Insights on transfer optimization: Because experience is the best teacher," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 51–64, 2017.

[192] Q. Shang, Y. Huang, Y. Wang, M. Li, and L. Feng, "Solving vehicle routing problem by memetic search with evolutionary multitasking," *Memetic Computing*, vol. 14, no. 1, pp. 31–44, 2022.

[193] L. Feng, L. Zhou, J. Zhong, A. Gupta, Y.-S. Ong, K.-C. Tan, and A. Qin, "Evolutionary multitasking via explicit autoencoding," *IEEE Transactions on Cybernetics*, vol. 49, no. 9, pp. 3457–3470, 2018.

[194] K. K. Bali, A. Gupta, Y.-S. Ong, and P. S. Tan, "Cognizant multitasking in multiobjective multifactorial evolution: Mo-mfea-ii," *IEEE Transactions on Cybernetics*, vol. 51, no. 4, pp. 1784–1796, 2020.

[195] T. Wei and J. Zhong, "Towards generalized resource allocation on evolutionary multitasking for multi-objective optimization," *IEEE Computational Intelligence Magazine*, vol. 16, no. 4, pp. 20–37, 2021.

[196] J. Zhong, L. Feng, W. Cai, and Y.-S. Ong, "Multifactorial genetic programming for symbolic regression problems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 11, pp. 4492–4505, 2018.

[197] M. Gong, Z. Tang, H. Li, and J. Zhang, "Evolutionary multitasking with dynamic resource allocating strategy," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 858–869, 2019.

[198] K. Qiao, K. Yu, B. Qu, J. Liang, H. Song, and C. Yue, "An evolutionary multitasking optimization framework for constrained multiobjective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 2, pp. 263–277, 2022.

[199] K. Chen, B. Xue, M. Zhang, and F. Zhou, "An evolutionary multitasking-based feature selection method for high-dimensional classification," *IEEE Transactions on Cybernetics*, 2020. Doi: 10.1109/TCYB.2020.3042243.

[200] J. Park, Y. Mei, S. Nguyen, G. Chen, and M. Zhang, "Evolutionary multitask optimisation for dynamic job shop scheduling using niched genetic programming," in *Proceedings of the Australasian Joint Conference on Artificial Intelligence*. Springer, 2018, pp. 739–751.

[201] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "A preliminary approach to evolutionary multitasking for dynamic flexible job shop scheduling via genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2020, pp. 107–108.

[202] F. Zhang, Y. Mei, S. Nguyen, K. C. Tan, and M. Zhang, "Task relatedness based multitask genetic programming for dynamic flexible job shop scheduling," *IEEE Transactions on Evolutionary Computation*, 2022.

[203] B. Khan, S. Hanoun, M. Johnstone, C. P. Lim, D. Creighton, and S. Nahavandi, "Multi-objective job shop scheduling using i-nsga-iii," in *Proceedings of the Annual IEEE International Systems Conference*. IEEE, 2018, pp. 1–5.

[204] R. Ojstersek, M. Brezocnik, and B. Buchmeister, "Multi-objective optimization of production scheduling with evolutionary computation: A review," *International Journal of Industrial Engineering Computations*, vol. 11, no. 3, pp. 359–376, 2020.

[205] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[206] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[207] F. Zhang, Y. Mei, and M. Zhang, "Evolving dispatching rules for multi-objective dynamic flexible job shop scheduling via genetic programming hyper-heuristics," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2019, pp. 1366–1373.

[208] J. C. Tay and N. B. Ho, "Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems," *Computers & Industrial Engineering*, vol. 54, no. 3, pp. 453–473, 2008.

[209] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Dynamic multi-objective job shop scheduling: A genetic programming approach," in *Automated scheduling and planning*. Springer, 2013, pp. 251–282.

[210] S. Shady, T. Kaihara, N. Fujii, and D. Kokuryo, "Evolving dispatching rules using genetic programming for multi-objective dynamic job shop scheduling with machine breakdowns," *Procedia CIRP*, vol. 104, pp. 411–416, 2021.

[211] D. Karunakaran, G. Chen, and M. Zhang, "Parallel multi-objective job shop scheduling using genetic programming," in *Proceedings of the Australasian Conference on Artificial Life and Computational Intelligence*. Springer, 2016, pp. 234–245.

[212] S. Salama, T. Kaihara, N. Fujii, and D. Kokuryo, "Multi-objective approach with a distance metric in genetic programming for job shop scheduling," *International Journal of Automation Technology*, vol. 16, no. 3, pp. 296–308, 2022.

[213] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "A coevolution genetic programming method to evolve scheduling policies for dynamic multi-objective job shop scheduling problems." in *IEEE Congress on Evolutionary Computation*, 2012, pp. 1–8.

[214] Y. Zhou and J. Yang, "Automatic design of scheduling policies for dynamic flexible job shop scheduling by multi-objective genetic programming based hyper-heuristic," *Procedia CIRP*, vol. 79, pp. 439–444, 2019.

[215] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Learning iterative dispatching rules for job shop scheduling with genetic programming," *The International Journal of Advanced Manufacturing Technology*, vol. 67, no. 1-4, pp. 85–100, 2013.

[216] A. Masood, Y. Mei, G. Chen, and M. Zhang, "Many-objective genetic programming for job-shop scheduling," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2016, pp. 209–216.

[217] ——, "A pso-based reference point adaption method for genetic programming hyper-heuristic in many-objective job shop scheduling," in *Proceedings of the Australasian Conference on Artificial Life and Computational Intelligence*. Springer, 2017, pp. 326–338.

[218] S. Nguyen, D. Thiruvady, M. Zhang, and K. C. Tan, "A genetic programming approach for evolving variable selectors in constraint programming," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 3, pp. 492–507, 2021.

[219] J. Park, S. Nguyen, M. Zhang, and M. Johnston, "Enhancing heuristics for order acceptance and scheduling using genetic programming," in

Asia-pacific conference on simulated evolution and learning. Springer, 2014, pp. 723–734.

[220] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Phenotype based surrogate-assisted multi-objective genetic programming with brood recombination for dynamic flexible job shop scheduling," in *Proceedings of IEEE Symposium Series On Computational Intelligence*. Springer, 2022.

[221] K. R. Escott, H. Ma, and G. Chen, "Genetic programming based hyper heuristic approach for dynamic workflow scheduling in the cloud," in *International Conference on Database and Expert Systems Applications*. Springer, 2020, pp. 76–90.

[222] Y. Yu, Y. Feng, H. Ma, A. Chen, and C. Wang, "Achieving flexible scheduling of heterogeneous workflows in cloud through a genetic programming based approach," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2019, pp. 3102–3109.

[223] W. B. Langdon and P. Treleaven, "Scheduling maintenance of electrical power transmission networks using genetic programming," *IEE Power Series*, pp. 220–237, 1997.

[224] D. Lynch, M. Fenton, S. Kucera, H. Claussen, and M. O'Neill, "Scheduling in heterogeneous networks using grammar-based genetic programming," in *Proceedings of the European Conference on Genetic Programming*. Springer, 2016, pp. 83–98.

[225] T. Saber, D. Fagan, D. Lynch, S. Kucera, H. Claussen, and M. O'Neill, "A multi-level grammar approach to grammar-guided genetic programming: the case of scheduling in heterogeneous networks," *Genetic Programming and Evolvable Machines*, vol. 20, no. 2, pp. 245–283, 2019.

[226] F. Seredyński, J. Koronacki, and C. Z. Janikow, "Distributed multiprocessor scheduling with decomposed optimization criterion," *Future Generation Computer Systems*, vol. 17, no. 4, pp. 387–396, 2001.

[227] T. Hildebrandt, D. Goswami, and M. Freitag, "Large-scale simulation-based optimization of semiconductor dispatching rules," in *Proceedings of the Winter Simulation Conference 2014*. IEEE, 2014, pp. 2580–2590.

[228] C. Pickardt, J. Branke, T. Hildebrandt, J. Heger, and B. Scholz-Reiter, "Generating dispatching rules for semiconductor manufacturing to minimize weighted tardiness," in *Proceedings of the winter simulation conference*. IEEE, 2010, pp. 2504–2515.

[229] H. Chen, G. Ding, S. Qin, and J. Zhang, "A hyper-heuristic based ensemble genetic programming approach for stochastic resource constrained project scheduling problem," *Expert Systems with Applications*, vol. 167, p. 114174, 2021.

[230] C. S. Pereira, D. M. Dias, M. M. Vellasco, F. H. F. Viana, and L. Martí, "Crude oil refinery scheduling: Addressing a real-world multiobjective problem through genetic programming and dominance-based approaches," in *Proceedings of the genetic and evolutionary computation conference companion*, 2018, pp. 1821–1828.

[231] C. S. Pereira, D. M. Dias, M. A. C. Pacheco, M. M. R. Vellasco, A. V. A. da Cruz, and E. H. Hollmann, "Quantum-inspired genetic programming algorithm for the crude oil scheduling of a real-world refinery," *IEEE Systems Journal*, vol. 14, no. 3, pp. 3926–3937, 2020.

[232] H. Lv, B. Wang, W. Zhang, and T. Li, "Genetic programming-based heuristic generation algorithm for steelmaking-continuous casting scheduling," in *Proceedings of the International Symposium on Computational Intelligence and Design*. IEEE, 2021, pp. 148–151.

[233] C. E. Nugraheni and L. Abednego, "On the development of hyper heuristics based framework for scheduling problems in textile industry," *International Journal of Modeling and Optimization*, vol. 6, no. 5, p. 272, 2016.

[234] J. Lin, L. Zhu, and K. Gao, "A genetic programming hyper-heuristic approach for the multi-skill resource constrained project scheduling problem," *Expert Systems with Applications*, vol. 140, p. 112915, 2020.

[235] L. Zhu, J. Lin, Y. Y. Li, and Z. J. Wang, "A decomposition-based multi-objective genetic programming hyper-heuristic approach for the multi-skill resource constrained project scheduling problem," *Knowledge-Based Systems*, vol. 225, p. 107099, 2021.

[236] J. Luo, M. Vanhoucke, J. Coelho, and W. Guo, "An efficient genetic programming approach to design priority rules for resource-constrained project scheduling problem," *Expert Systems with Applications*, vol. 198, p. 116753, 2022.

[237] H. Chen, G. Ding, J. Zhang, R. Li, L. Jiang, and S. Qin, "A filtering genetic programming framework for stochastic resource constrained multi-project scheduling problem under new project insertions," *Expert Systems with Applications*, vol. 198, p. 116911, 2022.

[238] H. Chen, J. Zhang, R. Li, G. Ding, and S. Qin, "A two-stage genetic programming framework for stochastic resource constrained multi-project scheduling problem under new project insertions," *Applied Soft Computing*, p. 109087, 2022.

[239] G. Pawiński and K. Sapiecha, "An efficient solution of the resource constrained project scheduling problem based on an adaptation of the developmental genetic programming," in *Recent Advances in Computational Optimization*. Springer, 2016, pp. 205–223.

[240] N. R. Sabar, A. Turky, and A. Song, "A genetic programming based iterated local search for software project scheduling," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018, pp. 1364–1370.

[241] A. Mahajan and M. S. Ali, "Superblock scheduling using genetic programming for embedded systems," in *Proceedings of the IEEE International Conference on Cognitive Informatics*. IEEE, 2008, pp. 261–266.

[242] X. Chen, R. Bai, R. Qu, and H. Dong, "Cooperative double-layer genetic programming hyper-heuristic for online container terminal truck dispatching," 2022. DOI: 10.1109/TEVC.2022.3209985.

[243] J. MacLachlan, Y. Mei, F. Zhang, and M. Zhang, "Genetic programming for vehicle subset selection in ambulance dispatching," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2022. DOI: 10.1109/CEC55065.2022.9870323.

[244] F. Zhang, Y. Chen, and Y. Chen, "Evolving constructive heuristics for agile earth observing satellite scheduling problem with genetic programming," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2018, pp. 1–7.

[245] C. Molnar, *Interpretable machine learning*. Lulu. com, 2020.

[246] E. Tjoa and C. Guan, "A survey on explainable artificial intelligence (xai): Toward medical xai," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 11, pp. 4793–4813, 2020.

[247] S. S. Malladi, A. L. Erera, and C. C. White III, "A dynamic mobile production capacity and inventory control problem," *Iise Transactions*, vol. 52, no. 8, pp. 926–943, 2020.

[248] M. J. Walter, D. J. Walker, and M. J. Craven, "Visualising population dynamics to examine algorithm performance," *IEEE Transactions on Evolutionary Computation*, 2022.

[249] S. Nguyen, M. Zhang, D. Alahakoon, and K. C. Tan, "People-centric evolutionary system for dynamic production scheduling," *IEEE Transactions on Cybernetics*, vol. 51, no. 3, pp. 1403–1416, 2019.

[250] W. Banzhaf, "Genotype-phenotype-mapping and neutral variation—a case study in genetic programming," in *Proceedings of the International Conference on Parallel Problem Solving from Nature*. Springer, 1994, pp. 322–332.

[251] R. E. Keller, W. Banzhaf, J. Koza, D. Goldberg, D. Fogel, and R. Riolo, "Genetic programming using mutation, reproduction and genotype-phenotype mapping from linear binary genomes into linear lalr (1) phenotypes," in *Koza, Goldberg, Fogel & Riolo, eds. Genetic Programming 1996: Proceedings of the First Annual Conference. MIT Press. ISSN*. Citeseer, 1996, pp. 1088–4750.

[252] T. Hu, M. Tomassini, and W. Banzhaf, "A network perspective on genotype–phenotype mapping in genetic programming," *Genetic Programming and Evolvable Machines*, vol. 21, no. 3, pp. 375–397, 2020.

[253] S. Nguyen, M. Zhang, and K. C. Tan, "Enhancing genetic programming based hyper-heuristics for dynamic multi-objective job shop scheduling problems," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2015, pp. 2781–2788.

[254] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Automatic programming via iterated local search for dynamic job shop scheduling," *IEEE Transactions on Cybernetics*, vol. 45, no. 1, pp. 1–14, 2015.

[255] S. Nguyen, Y. Mei, B. Xue, and M. Zhang, "A hybrid genetic programming algorithm for automated design of dispatching rules," *Evolutionary Computation*, vol. 27, no. 3, pp. 467–496, 2019.

[256] A. Farahani, S. Voghoei, K. Rasheed, and H. R. Arabnia, "A brief review of domain adaptation," *Advances in data science and information engineering*, pp. 877–894, 2021.

[257] B. Xu, Y. Mei, Y. Wang, Z. Ji, and M. Zhang, "Genetic programming with delayed routing for multiobjective dynamic flexible job shop scheduling," *Evolutionary Computation*, vol. 29, no. 1, pp. 75–105, 2021.

[258] J. Wang, Y. Liu, S. Ren, C. Wang, and S. Ma, "Edge computing-based real-time scheduling for digital twin flexible job shop with variable time window," *Robotics and Computer-Integrated Manufacturing*, vol. 79, p. 102435, 2023.