


Genetic Programming with Tabu List for Dynamic Flexible Job Shop Scheduling

Fangfang Zhang 

fangfang.zhang@ecs.vuw.ac.nz

Centre for Data Science and Artificial Intelligence & School of Engineering and
Computer Science, Victoria University of Wellington,
PO Box 600, Wellington 6140, New Zealand

Mazhar Ansari Ardeh 


mazhar.ansari.ardeh@gmail.com

Wolt, Berlin, Germany

Yi Mei 

yi.mei@ecs.vuw.ac.nz

Centre for Data Science and Artificial Intelligence & School of Engineering and
Computer Science, Victoria University of Wellington,
PO Box 600, Wellington 6140, New Zealand

Mengjie Zhang 

mengjie.zhang@ecs.vuw.ac.nz

Centre for Data Science and Artificial Intelligence & School of Engineering and
Computer Science, Victoria University of Wellington,
PO Box 600, Wellington 6140, New Zealand

Abstract

Dynamic flexible job shop scheduling (DFJSS) is an important combinatorial optimisation problem, requiring simultaneous decision-making for machine assignment and operation sequencing in dynamic environments. Genetic programming (GP), as a hyper-heuristic approach, has been extensively employed for acquiring scheduling heuristics for DFJSS. A drawback of GP for DFJSS is that GP has weak exploration ability indicated by its quick diversity loss during the evolutionary process. This paper proposes an effective GP algorithm with tabu lists to capture the information of explored areas and guide GP to explore more unexplored areas to improve GP's exploration ability for enhancing GP's effectiveness. First, we use phenotypic characterisation to represent the behaviour of tree-based GP individuals for DFJSS as vectors. Then, we build tabu lists that contain phenotypic characterisations of explored individuals at the current generation and across generations, respectively. Finally, newly generated offspring are compared with the individuals' phenotypic characterisations in the built tabu lists. If an individual is unseen in the tabu lists, it will be kept to form the new population at the next generation. Otherwise, it will be discarded. We have examined the proposed GP algorithm in nine different scenarios. The findings indicate that the proposed algorithm outperforms the compared algorithms in the majority of scenarios. The proposed algorithm can maintain a diverse and well-distributed population during the evolutionary process of GP. Further analyses show that the proposed algorithm does cover a large search area to find effective scheduling heuristics by focusing on unseen individuals.

Keywords

Genetic Programming, Tabu List, Exploration Ability, Scheduling Heuristics, Dynamic Flexible Job Shop Scheduling.

1 Introduction

Job shop scheduling is an important combinatorial optimisation problem which optimises machine resources to process jobs that consist of a number of operations (Manne, 1960; Zhang et al., 2023a). In traditional job shop scheduling, all operations are allocated to particular machines in advance, and we only need to make *operation sequencing* decision to choose the next operation to be processed on idle machines (Hart and Sim, 2016; Gere Jr, 1966). Dynamic flexible job shop scheduling (DFJSS) is a variation of job shop scheduling (Zhou et al., 2020; Xu et al., 2022; Destouet et al., 2023) with a number of valuable real world applications such as fog computing (Xu et al., 2023b) and order picking in warehouse (D’Haen et al., 2023). For DFJSS, we are required to make two decisions: firstly, determining the *machine assignment* to allocate operations to machines either upon the arrival of new jobs or when an operation is completed and its subsequent operation becomes ready for processing, and secondly, deciding the *operation sequencing* when a machine becomes idle and there are operations waiting in its queue.

Exact methods such as integer linear programming (Liu et al., 2021; Bülbül and Kaminsky, 2013), can find optimal solutions for job shop scheduling. However, they are normally only applicable to small scale and static problems due to their inefficiency. Heuristic algorithms such as genetic algorithms (Davis, 2014; Biegel and Davern, 1990), have been successfully used to tackle large scale job shop scheduling problems efficiently. Nevertheless, their efficiency is compromised in dynamic scenarios, primarily due to the inefficiency in rescheduling when confronted with dynamic events. Scheduling heuristics like shortest processing time (SPT), have been widely used for job shop scheduling by prioritising the candidate operations on machines (Kaban et al., 2012; Canbolat and Gundogar, 2004). However, they are normally manually designed, a process that is both time-consuming and reliant on expertise which may not always be readily available. Genetic programming (GP), has been widely used to automatically learn scheduling heuristics for DFJSS (Zhang et al., 2023b; Jaklinović et al., 2021; Zhu et al., 2023). In particular, a routing rule is employed for machine assignment, and concurrently, a sequencing rule is acquired for operation sequencing. As we know, as an evolutionary computation algorithm, exploration ability is important for GP search (Braune et al., 2022; Salama et al., 2022). From literature (Nicolau and Fenton, 2016; Zhang et al., 2022d; Sitahong et al., 2022), and our preliminary results, we find that there is an obvious exploration decrease with many seen individuals during the evolutionary process of GP, which is more likely to limit its search/exploration ability. However, the study of interacting with population individuals to guide the exploration of GP during the evolutionary process with tree-based representation in DFJSS is limited.

The fundamental concept behind tabu search involves engaging in local search when confronted with a local optimum by permitting non-improving moves. The prevention of cycling back to previously visited solutions is achieved through the utilisation of memories referred to as *tabu lists* (Prajapati et al., 2020). Inspired by tabu search (Li and Gao, 2016), this paper aims to improve the exploration of GP by controlling the search space with tabu lists, where the tabu list stores the information of all the explored GP individuals during the GP search process. This paper encourages GP to search more on unexplored areas that have not been seen in the tabu lists. If a newly generated individual is not seen in the tabu list, it represents an unexplored area. There are four research challenges for using tabu lists to guide the search of GP with tree-based programs for DFJSS. First, how to represent the tree-based GP individuals in DFJSS for building a tabu list? A proper way to represent the variable-length GP

individuals is critical for the success of using tabu list to memory visited individuals to guide the GP search. Second, how to improve the computational efficiency of using tabu list, especially when the size of the tabu list becomes larger and larger generation by generation? Third, what is an effective way of guiding the search of GP with a tabu list? Fourth, how will the search with tabu list affect the exploration of GP, and also its performance of learning scheduling heuristics for DFJSS? It is not clear what is a good pressure to guide GP to search more on unexplored areas according to the tabu list.

To tackle these challenges, first, we introduce to use phenotypic characterisation (Hildebrandt and Branke, 2015) which is a vector to represent the behaviour of GP individuals. These vectors of all individuals from previous generations are added to the tabu list as a memory for representing explored areas/individuals of GP. Second, we use Hash Table (Maier et al., 2019) to store key-value pairs for checking the tabu list. The foundational concept of a hash table revolves around utilising a hash function to determine an index within an array of buckets or slots, facilitating the retrieval of the desired value. It is an efficient way to check whether an individual is explored or not. Third, specialised crossover and mutation operators are designed to encourage GP to explore unexplored areas by generating individuals which are not seen in the tabu list. Fourth, we conduct parameter sensitivity analyses of the pressure level to explore unexplored areas, thus investigating how the proposed algorithm affects the diversity of population, and also the performance of GP to learn scheduling heuristics for DFJSS.

The goal of this paper is to develop an effective GP algorithm to learn scheduling heuristics by improving the exploration of GP with tabu lists. The proposed algorithm is expected to cover a wide range of explored areas to give more changes to find better scheduling heuristics. In particular, this paper's key contributions are outlined as follows:

1. We have introduced to use phenotypic characterisation to represent the behaviour of variable-length GP individuals on decision marking for DFJSS to build tabu lists. The tabu list stores the explored areas/individuals by recording phenotypic characterisations of all visited individuals from the perspective of GP's phenotype point of view. This paper provides a foundational study of incorporating tabu list into GP algorithm with variable-length representation for DFJSS. This idea is generic and applicable for other problems if proper phenotypic characterisations can be designed according to the problems.
2. We have developed an effective GP algorithm for DFJSS with the built tabu list to guide GP to search more on unexplored areas. Specifically, the search towards unexplored areas is managed via newly generated individuals for the next generation. The proposed algorithm provides good insights on investigating the exploration of GP.
3. We have shown that the proposed GP algorithm with tabu lists can learn highly competitive scheduling heuristics for DFJSS. The effectiveness of the proposed algorithm is reflected by well-distributed individuals in the search space, and good preservation of population diversity. The corresponding analyses benefit to the understanding of the process of GP to learn scheduling heuristics for DFJSS.
4. Further analysis and discussions show that the newly generated offspring of the proposed algorithm contains more unseen individuals than seen individuals to build the next generation. The results also show that it becomes harder and harder to generate unseen individuals along with generations, which is consistent with

our intuition since more and more individuals have been explored. These findings indicate that it is worth paying attention on unseen individuals, which suggests a possible direction of improving the performance of the GP algorithm.

The subsequent sections of this paper are structured as follows: Section 2 provides a review of the literature. Section 3 offers detailed explanations of the proposed algorithm. The experimental design is outlined in Section 4. The findings and discussions are presented in Section 5. Additional analyses are carried out in Section 6. Finally, Section 7 serves as the conclusion for this paper.

2 Background

2.1 Dynamic Flexible Job Shop Scheduling

In DFJSS, a set of jobs $\mathcal{J} = J_1, J_2, \dots, J_n$ needs to be processed on a set of machines $\mathcal{M} = M_1, M_2, \dots, M_m$. Each job has a list of ordered operations, and the number of operations is job-dependent. The completion of a job signifies the successful processing of all its operations. In DFJSS, machine resources are flexible, allowing each operation to be processed on multiple machines with different processing efficiency (Brucker and Schlie, 1990). The candidate machines of an operation are operation-dependent, and different operations may have different set of machines to process them. However, each operation is assigned to a specific candidate machine, and its processing time is contingent upon the machine handling the operation. Taking a practical example, an aircraft manufacturing plant produces a variety of components (jobs) for jet engines. Each component (job) must undergo a series of operations such as cutting, drilling, grinding, heat treatment, and inspection. However, the plant has multiple machines capable of performing the same operation. For example, Heat Treatment Oven X and Oven Y can both handle heat treatment, but Oven X is more energy-efficient. This paper concentrates on a dynamic event: the dynamic and stochastic arrival of new jobs (Zhang et al., 2020c; Durasevic and Jakobovic, 2018). This choice is motivated by the prevalence of this dynamic event in real-life scenarios. This suggests that information about jobs remains unknown until their actual arrival on the shop floor. The primary constraints of DFJSS are delineated as follows.

- A machine is capable of processing only one operation at a time.
- Each operation is restricted to being processed by only one of its candidate machines.
- Processing of an operation cannot commence until all its preceding operations have been completed.
- The processing of an operation cannot be halted or paused until it reaches completion, once started.

This paper addresses three frequently employed objectives, enumerated as follows:

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

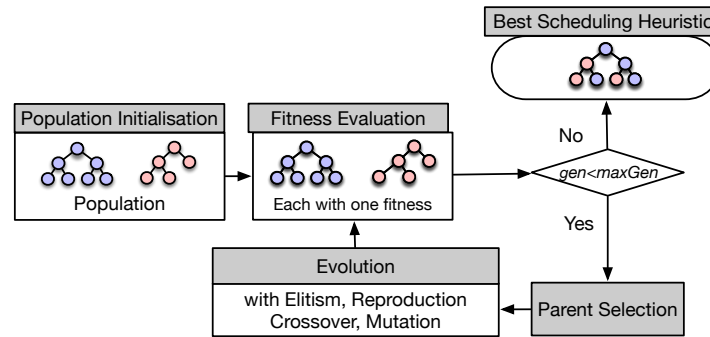


Figure 1: The flowchart illustrating the process of GP to evolve scheduling heuristics for DFJSS.

where r_j denotes the release time of J_j , C_j represents the completion time of job J_j , d_j signifies the due date of J_j , w_j reflects the weight (importance) assigned to job J_j , and n represents the total number of jobs. Note that the release time is the ready time of starting processing a job, which is the same as arrival time in this paper, i.e., a job will be ready to be processed once it arrives. The completion time is the time gap between finish time and release time. The due date is the time that a job should be finished.

2.2 Genetic Programming for DFJSS

Figure 1 depicts the flowchart representing GP for acquiring scheduling heuristics in DFJSS, where the primary steps align with typical GP procedures. GP emulates the evolutionary mechanism in nature to enhance offspring generation over successive generations, encompassing four key processes: initialisation, evaluation, parent selection, and evolution (Langdon and Poli, 2013). GP commences with a set of randomly initialised individuals. The effectiveness of each GP individual is assessed using DFJSS instances (simulations) during the evaluation phase. If the stop criterion is not met, parent selection is carried out through genetic operators, i.e., elitism, reproduction, crossover, and mutation to generate offspring for the subsequent generation. Conversely, if the stop criterion is satisfied, the GP algorithm outputs the best-found scheduling heuristic for the given DFJSS problem.

2.2.1 Representation

When it comes to acquiring scheduling heuristics for DFJSS, GP with a multi-tree representation demonstrates its superiority by simultaneously evolving the routing rule for machine assignment and the sequencing rule for operation sequencing (Zhang et al., 2018, 2022c). Figure 2 illustrates a GP individual exemplifying both the routing rule and the sequencing rule for DFJSS. These rules collaborate to produce schedules. The routing rule gives precedence to machines using the criterion $NIQ + WIQ * MWT$. Here, NIQ and WIQ represent the count of operations, and the needed time for a machine to complete all operations in the queue of a machine, respectively. MWT indicates the time required for a machine to finish the currently processing operation. Meanwhile, the sequencing rule prioritises operations based on PT / W , where PT stands for the needed processing time of each operation, and W reflects the priority of the operation.

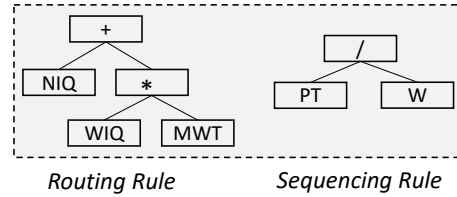


Figure 2: An example of a GP individual equipped with a routing rule and a sequencing rule designed for DFJSS.

Table 1: An illustration of using the routing rule $NIQ + WIQ * MWT$ for decision-making at a decision point with three machines.

Decision Situation	Candidate Machines	Features			Priority Values	Selected Machine
		(NIQ	WIQ	MWT)		
1	M_1	100	200	3	700	M_2
	M_2	30	100	5	<u>530</u>	
	M_3	4	50	100	<u>5004</u>	

2.2.2 Scheduling Heuristics for Machine Assignment and Operation Sequencing

Scheduling heuristics work as priority functions to prioritise operations or machines, which are functionally easy to understand. There are two categories of decision points, namely, “routing decision points” and “sequencing decision points”. Routing decision points correspond to cases when a new job arrives and its first operation is ready for processing, or when an operation is completed and its succeeding operation becomes ready for processing. The sequencing decision points correspond to the time when a machine becomes idle, and its queue is not empty. To illustrate the routing decision process, Table 1 provides an example of calculating the machine priority values of machines for assigning a ready operation. In this example, three machines (M_1 , M_2 , and M_3) can process the ready operation. The priority values for these machines are computed using the routing rule from Figure 2 and the values of the relevant features. The calculated priority values for M_1 , M_2 , and M_3 are 700, 530, and 5004, respectively. Consequently, the machine with the highest priority (indicated by the smallest value and underlined) is selected to allocate the operation, which in this case is M_2 .

2.3 Related Work

Exploration plays a crucial role in the success of GP algorithms, as it helps explore a broader search space and can help the discovery of more effective solutions (Vanneschi et al., 2014). To the best of our knowledge, the studies of GP with tabu list is limited. This section discusses some related studies on GP diversity which are related to the exploration ability of GP algorithms. Maintaining a proper diversity level of the GP population can help with its exploration ability (Kelly et al., 2019).

2.3.1 Diversity in GP

Unlike traditional evolutionary algorithms with vector based representations, it is not straightforward to measure the diversity of GP with tree-based representation. Researchers have proposed various diversity measures to quantify the differences among individuals in a GP population (Burke et al., 2004; Jackson, 2010; Burke et al., 2002). Common diversity measures include genetic diversity, phenotypic diversity, and behavioural diversity. These metrics help GP practitioners understand and track the level

of diversity in the GP populations. To enhance the diversity of GP, researchers have developed a variety of techniques. These include diversity maintenance strategies, such as crowding (Galván and Schoenauer, 2019), speciation (Juárez-Smith et al., 2019), and fitness sharing (Burke et al., 2004), which encourage the survival of diverse individuals during the evolutionary process. Additionally, methods like niching (De Aruda Pereira et al., 2014) and island models (Ono et al., 2019) involve maintaining population to promote diversity.

2.3.2 GP Diversity for Job Shop Scheduling

Normally, the GP diversity for job shop scheduling is measured by considering the differences of GP individuals in decision marking. The idea of using phenotypic characterisations to represent the behaviour of GP individuals was first proposed in (Hildebrandt and Branke, 2015) for dynamic job shop scheduling. Simply speaking, the phenotypic characterisation is a vector to reflect the behaviour of a GP individual by looking at the ranks of chosen operations at multiple sequencing decision points. This idea was further extended to measure the behaviour of GP individuals for DFJSS by constructing phenotypic characterisations with machine assignment and operation sequencing decisions simultaneously (Zhang et al., 2021b,c, 2022a).

Existing related studies mainly focus on how to select parents to improve the population diversity of GP for job shop scheduling. For example, an archive was developed to save promising individuals from different generations, and individuals in the archive were given probabilities to be parents to generate individuals in the later generations (Xu et al., 2021). GP individuals were grouped based on their phenotypic characterisations, and the parents from different groups (different behaviour) were encouraged to generate offspring with crossover for the next generation (Xu et al., 2022). The performance of GP individuals in different cases was also utilised to choose parents for genetic operators in (Xu et al., 2022, 2023a,b).

Although these studies show an increase of population diversity during the evolutionary process, the diversity information are only used as a metric to show the effect of the proposed algorithms rather than guiding the exploration of GP for DFJSS. In addition, generally speaking, increasing the population size can potentially improve the exploration effectiveness of GP, since the population may cover more search areas. However, this is also related to the search space of a problem. If the search space of a problem is small, increasing population size will not help much. In addition, based on our preliminary investigations, a big issue of GP for DFJSS is that the population diversity reduces dramatically at the very early stage of GP. This is the research problem we focus in this paper. To the best of our knowledge, the studies of improving the effectiveness of GP by guiding the GP exploration to unseen search space for DFJSS are rare. In addition, most studies of improving GP's exploration capabilities such as improved parent selection methods for generating offspring, or increasing the mutation rate, were designed to improve the exploration ability of GP at a current generation, ignoring the explored individuals in all previous generations (i.e., the whole evolutionary process).

3 Proposed GP Algorithm with Tabu List for DFJSS

3.1 Framework of the Proposed GP Algorithm with Tabu List

Figure 3 illustrates the flowchart depicting the proposed GP algorithm. The proposed algorithm starts with population initialisation. We introduce the use of phenotypic characterisations (PC) (Hildebrandt and Branke, 2015) of GP individuals to build tabu lists. The PC of a GP individual is represented as a vector containing the ranks of ex-

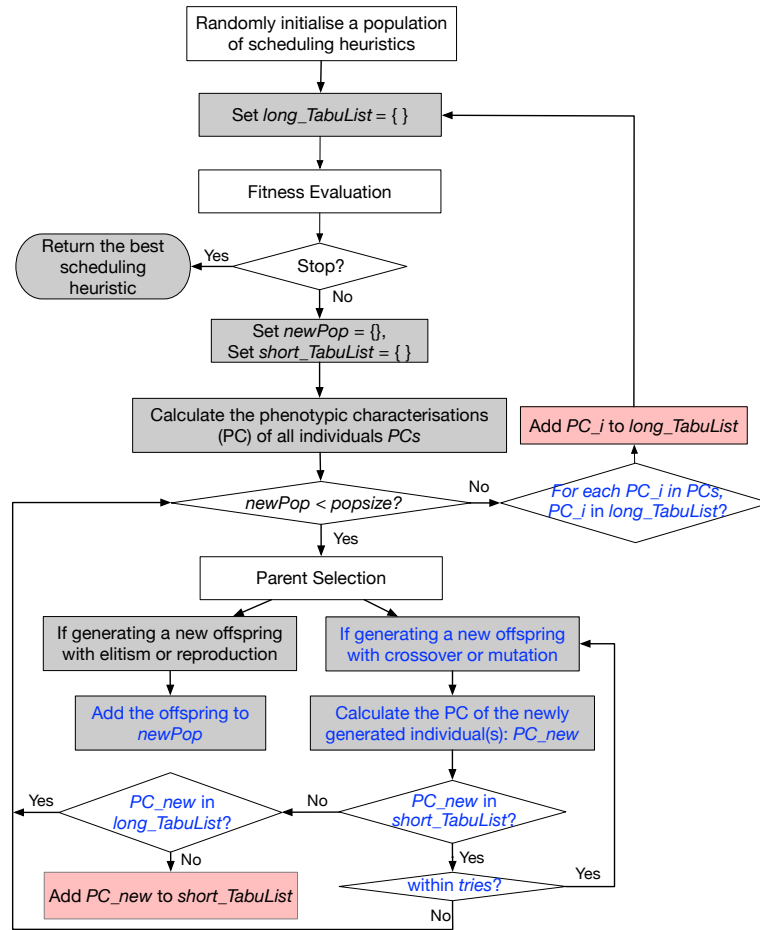


Figure 3: The flowchart of the proposed GP algorithm with tabu list.

amined machines and operations. This vector serves as a representation of the individual's decision-making behavior in response to a set of routing and sequencing decision situations. Each dimension of the PC aligns with the rank of the foremost-prioritised machine or operation determined by the examined routing or sequencing rule, as dictated by corresponding reference rules. We use *long_TabuList* to memorise all explored areas by recording the PCs of all seen individuals across generations so far, while we use *short_TabuList* to memorise the explored area at the current generation.

The proposed GP algorithm with tabu list to enhance the exploration ability is conducted by examining the newly generated individuals via crossover and mutation. The steps of updating *short_TabuList* and *long_TabuList* are highlighted with pink background. Other main steps are highlighted with grey background, and important steps are highlighted in blue. For each offspring generated with crossover and mutation, we will calculate the PC of this individual *PC_new*. If *PC_new* is not seen in *short_TabuList*, then we will further check if *PC_new* is also not seen by *long_TabuList*. If *PC_new* does not exist in *long_TabuList* either, the corresponding individual is considered as an unexplored area to be added to the next generation. In

Table 2: An illustration of computing the phenotypic characterisation with two decision scenarios, each involving three candidate machines, for a routing rule.

Decision Situation	Ranking by the reference rule, i.e., WIQ	Ranking by a routing rule	PC_i
1 (M_1)	1	3	2
1 (M_2)	3	2	
1 (M_3)	<u>2</u>	<u>1</u>	
2 (M_1)	2	2	3
2 (M_2)	1	3	
2 (M_3)	<u>3</u>	<u>1</u>	

PC_i represents the i^{th} dimension of phenotypic characterisation.

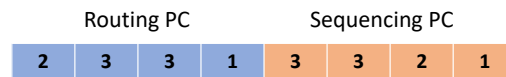


Figure 4: An illustration of the phenotypic characterisation of a GP individual for DFJSS with routing PC and sequencing PC represented as an 8-dimensional vector derived with four routing decision situations and four sequencing decision situations.

addition, PC_{new} will be added to *short.TabuList* to record explored individuals at the present generation. If the newly generated offspring are seen, the current parents can be used to generate new offspring for a check again with a limited number of times represented by *tries*. Note that after finishing the new population generation, we start to use update *long.TabuList* with PCs of the parent population. For any PC_i that does not appear in *long.TabuList*, we will add it into *long.TabuList*. Since elitism and reproduction aim to keep good individuals to the next generation directly to keep a good trend of the algorithm improvement, the corresponding individuals will not checked with *long.TabuList* or *short.TabuList*. The details of such design are shown in the following subsections.

3.2 Tabu List

3.2.1 Building a Tabu List

A tabu list contains PCs of GP individuals. With two examined decision situations, Table 2 provides an illustration of the process of computing a routing rule's phenotypic characterisation. PC_i denotes the i^{th} dimension of phenotypic characterisation, reflecting the rule's behaviour in decision situation i . We initially give ranks for candidate machines by the reference routing rule "least work in the queue" represented as WIQ. The routing rule selects M_3 in the first decision situation, resulting in PC_1 being the rank of M_3 by WIQ, which is 2. Similarly, PC_2 is 3. A sequencing rule can obtain its phenotypic characterisation in a similar manner. In this paper, we use SPT which represents shortest processing time as the reference sequencing rule. Given that our GP algorithm with a multi-tree representation (as depicted in Figure 2) to concurrently evolve the routing rule and the sequencing rule for DFJSS, we concatenate the phenotypic characterisations of both rules into the phenotypic characterisation of a GP individual. An illustration of the phenotypic characterisation for a GP individual in DFJSS represented as an 8-dimensional vector (comprising routing PC and sequencing PC), is depicted in Figure 4.

3.2.2 Search Efficiency with Tabu List

As discussed before, *short.TabuList* keeps the unique PCs of the individuals at the current generation, where the maximal size of *short.TabuList* is *popsiz*e, i.e., the population size. *long.TabuList* saves the unique PCs of the individuals across all previous generations, where its maximal size of *popsiz*e**numGenerations*, where *numGenerations* is the number of generations already gone through. For a newly generated individual, the PC of the individual will be compared with all PCs in *short.TabuList* and *long.TabuList*, and the maximal number of distance calculations among PCs is about *popsiz*e*(*numGenerations* + 1). Along with generations, this computational cost becomes higher and higher due to the increase of PCs in the tabu lists. An efficient searching mechanism is needed when checking whether a GP individual is seen or not. There are two designs for this purpose.

- Using hashing techniques for *short.TabuList* and *long.TabuList*: The search efficiency of a data structure refers to the speed with which the structure can locate a particular element or information within it. This paper proposes to use a hash table to construct *short.TabuList* and *long.TabuList*. With a properly implemented hash function, hash tables enable direct access to elements based on their keys. This direct mapping allows for quick search operations, making hash tables suitable for fast retrieval of data.
- Searching *short.TabuList* first, and then looking for *long.TabuList* if needed. If an individual has been seen in *short.TabuList*, we can stop the search and no need to check *long.TabuList* anymore.

To add an individual to the list, in our approach, first the phenotypic representation of each GP tree is calculated, and then the phenotypic representation is hashed into 32-bit integer and then stored into the tabu list. So, to store a tree, only 32 bits of memory is needed. Accordingly, considering our GP setup of 100 generations of evolution with a population size of 500 individuals, at most $(100 - 1) \times 500$ individuals are generated before the final generation. So, in the worst case scenario, $(100 - 1) \times 500 \times 2 \times 32$ bits of memory are needed to store the whole tabu list, i.e., note that each GP individual has two trees in this paper. This is equivalent to 387 megabytes (i.e., $(100-1) * 500 * 2 * 32 / (8 * 1024)$) of memory, which is easily accessible in modern computing devices. Additionally, during the evolutionary process, our proposed method searches the tabu list frequently to check if new individuals are inside the list or not. For this, the new individuals are hashed first, and then the hashed value is looked up in a hash table that stores the tabued items with a time complexity of $O(1)$. So, in essence, searching the tabu list can be performed very efficiently during the evolutionary process and will not impose any challenges.

3.3 Generating Unseen Individuals

How to generate unseen individuals is an important step for using the built tabu lists. Figure 5 shows an example of generating offspring for the next generation via crossover with two parents. For traditional crossover, one subtree is randomly selected for each parent highlighted by the dotted circle. The selected subtrees are swapped to generate two offspring. We can see that *Offspring₁* and *Offspring₂* are originated from *Parent₁* and *Parent₂*, respectively. In other words, *Offspring₁* will be close to *Parent₁*, and *Offspring₂* performs close to *Parent₂* (Zhang et al., 2022b). With multi-tree representation, there are two types of rules in a GP individual, as shown in Figure 2, and we

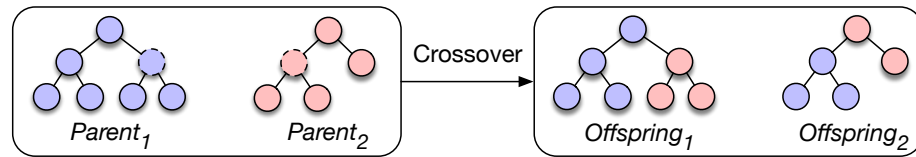


Figure 5: An example of generating offspring for the next generation via crossover with two parents, each parent having one tree.

only choose one type of tree to perform crossover every time, either the routing rule or the sequencing rule, and keep the other rule unchanged (Zhang et al., 2018). This paper only keeps unseen offspring when both $Offspring_1$ and $Offspring_2$ are unseen *simultaneously*. More discussions on this design are provided in Section 6.3. For mutation, after choosing one parent, one of the two trees as shown in Figure 2 will be selected, and a newly created subtree will be used to replace a randomly chosen subtree of the selected tree. If the newly generated offspring by crossover and mutation is not unseen in *short_TabuList* and *long_TabuList*, it will be kept to form the next generation. For crossover and mutation, the proposed algorithm tries to generate unseen individuals within a limited number of *tries* which controls the pressure to find unseen individuals. It is noted that the crossover and mutation operators do not generate unseen offspring directly, but the proposed algorithm only keeps unseen offspring into the next generation by filtering out the seen offspring via built tabu list.

3.4 Summary

The proposed algorithm suggests searching unexplored areas via looking for unseen individuals to learn scheduling heuristics for DFJSS within a limited effort represented by *tries*. The new population for the next generation contains seen individuals and unseen individuals. The seen individuals in the new population are from elitism and reproduction, and the kept newly generated seen individuals due to the failure to get unseen individuals after the limited of *tries*. We expect the proposed algorithm to have a relatively high ratio of unseen individuals than seen individuals for a good exploration ability. In addition, the proposed algorithm is expected to search in a broad search space to find effective scheduling heuristics, which will increase the population diversity.

4 Experiment Design

4.1 Simulation Model

Referring to established DFJSS instances (Tay and Ho, 2008; Nguyen et al., 2018) and adhering to the configurations in (Zhang et al., 2021b), the instances employed in this paper assume the processing of 5000 jobs using 10 machines. New jobs continuously enter the system in accordance with a Poisson process having a rate of λ . The number of operations for a job and the number of accessible machines for each operation is within a range of 1 and 10. An operation's processing time is determined through a range from 1 to 99 uniformly. A job's due date is related to its processing time, i.e., $1.5 \times \text{processing time}$. 20%, 60%, and 20% of the jobs are with weight values of 1, 2, and 4 to indicate their importance, respectively (Hildebrandt and Branke, 2015). Following the work in (Nguyen et al., 2017), we use a unique training instance at each generation which is implemented by changing a new random seed for the simulation to improve the generalisation of learned scheduling heuristics.

The utilisation level (p) plays a crucial role in simulating diverse DFJSS scenarios (Nguyen et al., 2015). It represents a machine's proportion time to be occupied, a factor adjusted with the parameter λ for simulating the Poisson process. The computation of the utilisation level is outlined in Equation (1), where μ denotes the average processing time of the machines, and P_M represents the probability of a job being processed on a machine. For instance, if each job consists of two operations, P_M would be 2/10. A higher utilisation level typically results in a more active and occupied job shop.

$$\lambda = \mu * P_M / p \quad (1)$$

For assessing the performance of algorithms in a stable state, the initial 1000 jobs are designated for a warm-up purpose and excluded from calculating the objective values. The result collection in this study focuses on the subsequent 5000 jobs. The simulation concludes upon the completion of the 6000th job.

For the training process, following the suggestion in (Hildebrandt et al., 2010), to improve the generalisation ability of learning rules and improve training efficiency, we generate one problem instance for every generation to evaluate the quality of GP individuals. The training instance is changed/rotated across different generations by using different random seed for the simulation. For the test, we apply the learned rules on 50 unseen instances, and report the average objective value of the 50 unseen instances as the test performance for each pair of rules. As we can see, a set of problem instances is used to train the routing rule and the sequencing rule, and the learned pair of the routing rule and the sequencing rule is used to test on unseen instances. The routing rule and the sequencing rule are tailored for a class of problem instances. Note that in traditional machine learning, normally the training instances have the same data distributions with the test instances, which is also the case in this paper. The learned models of the training is not expected to perform well on test with different problem distributions. However, it is possible to learn scheduling rules from training instances, and adapt them to test instances with different distributions from training instances. We will explore this more with adaptation techniques in our future work.

4.2 Comparison Design

We examine three distinct objectives: mean-flowtime (referred to as Fmean), mean-tardiness (referred to as Tmean), and mean-weighted-tardiness (referred to as WT-mean). Additionally, we consider three utilisation levels, i.e., 0.75, 0.85, and 0.95 to generate various DFJSS scenarios (Zhang et al., 2019, 2021a). The proposed algorithm is examined in nine scenarios, and each scenario is represented by <objective, utilisation level> such as <Fmean, 0.75>. Note that the nine scenarios are independent, the pair of the routing rule and the sequencing rule is trained on problem instances with different objectives and utilisation levels in different scenarios. The GP algorithm is used as the baseline algorithm. To verify the effectiveness of the proposed GP with tabu list, named tabuGP, tabuGP is compared with GP. In addition, GP and tabuGP is also compared with manually designed rules that are popularly used for scenarios. Specifically, for all scenarios, we use WIQ to allocate an operation to a machine with smallest workload. For scenarios with objective mean-flowtime, we use shortest processing time (PT) as the sequencing rule. For scenarios with objective mean-tardiness and mean-weighted-tardiness, we use ATC rule, i.e., apparent tardiness cost, and WATC rule, i.e., weighted apparent tardiness cost, as the sequencing rule, respectively (Chen and Matis, 2013).

Table 3: The terminal set.

Type	Notation	Description
machine-related	NIQ	The number of operations in the queue of a machine
	WIQ	Work in the queue of a machine, i.e., the needed time to process all operations
	MWT	Machine waiting time, i.e., needed time to finish the currently processing operation
	PT	Processing time of an operation on a specified machine
job-related	NPT	Median processing time (estimated) for the next operation of a current operation
	OWT	The waiting time of an operation in the queue of a machine
	WKR	Median amount of work (estimated) remaining for a job
	NOR	The number of operations remaining for a job
	W	Weight of a job
	TIS	Time of a job in system

Table 4: The parameter settings for evolving scheduling heuristics with GP.

Parameter	Value
Size of Population	500
The count of elite individuals	10
Method for parent selection	Tournament selection with size 5
Rate of Reproduction/Crossover/Mutation	5%/80%/15%
Initialising population method	ramped-half-and-half
Minimum / maximum of initial depth	2 / 6
Maximal depth of individuals	8
Selection rate of terminal / non-terminal	10% / 90%
The maximal number of generations	100

4.3 Parameter Settings

The terminals of GP are derived from the features of the job shop, as outlined in (Zhang et al., 2021c), which can be found in Table 3. This paper uses the same terminal set for learning both the routing rule and the sequencing rules (Zhang et al., 2021b, 2020a,b, 2022e). We have {+, -, *, /, max, min} as function set, and each function in the set has two arguments. The division function “/” is implemented as a protected division, yielding one if divided by zero. The function max and min containing two arguments, produce the maximum and minimum of the provided values, respectively. The rest of parameter settings for GP, as recommended in (Koza and Poli, 2005), are presented in Table 4.

In line with the recommendation in (Zhang et al., 2021c), we select 20 routing decisions and 20 sequencing decisions during an extensive simulation to obtain GP individuals’ phenotypic characterisations. Consequently, the dimension of an individual’s phenotypic characterisation is set to 40. The decision situations remain constant for calculating the phenotypic characterisations of all individuals. It’s important to note that the number of examined decisions’ operations and machines is with a fixed number which is 7 in this paper to ensure comparability of phenotypic characterisations among GP individuals.

5 Results and Discussions

We employ Friedman’s test and the Wilcoxon rank-sum test, both with a significance level of 0.05, to assess the performance of algorithms across 30 independent runs. The “Average Rank” derived from Friedman’s test provides an average rating for the algorithm across all scenarios. In the subsequent results, symbols such as “↑”, “↓”, and “≈” signify statistical significance, indicating that the corresponding result is significantly

F. Zhang, M. Ardeh, Y. Mei, and M. Zhang

Table 5: The mean (standard deviation) of fitness obtained on training instance of *tabuGP* with different numbers of tries to find unseen offspring for the next generation according to 30 independent runs in nine scenarios.

Scenario	TabuGP5	TabuGP10	TabuGP15	TabuGP20	TabuGP25	TabuGP30	TabuGP35	TabuGP40	TabuGP45	tabuGP50
<Fmean, 0.75>	339.40(1.32)	339.42(1.49)	338.84(1.61)	338.76(1.43)	338.27(0.83)	338.52(1.06)	338.60(1.17)	338.97(1.67)	339.07(1.52)	338.84(1.32)
<Fmean, 0.85>	392.80(3.86)	392.14(2.38)	392.54(3.06)	392.57(2.89)	391.41(1.88)	391.66(2.29)	392.11(2.18)	391.57(1.38)	390.97(1.13)	392.63(3.41)
<Fmean, 0.95>	571.01(10.69)	568.38(7.02)	569.59(10.38)	568.59(6.71)	568.56(6.90)	567.49(5.58)	567.50(6.70)	566.09(6.67)	567.89(5.90)	568.01(7.24)
<Tmean, 0.75>	12.64(0.94)	12.52(0.38)	12.52(0.36)	12.51(0.26)	12.57(0.68)	12.48(0.35)	12.48(0.31)	12.57(0.78)	12.52(0.55)	12.72(0.82)
<Tmean, 0.85>	40.81(0.85)	40.94(1.22)	40.78(0.90)	41.07(1.46)	40.61(0.80)	40.60(0.70)	41.08(1.94)	40.44(0.78)	41.01(1.71)	40.64(0.96)
<Tmean, 0.95>	190.69(6.80)	189.43(5.36)	187.64(4.66)	189.50(5.29)	188.69(7.00)	188.06(5.23)	188.73(5.95)	188.91(5.08)	187.81(4.52)	185.95(3.49)
<WTmean, 0.75>	25.45(1.61)	25.30(1.01)	25.29(0.64)	25.25(0.79)	25.13(0.50)	25.28(1.05)	25.06(0.52)	25.51(1.38)	25.25(1.16)	25.44(1.60)
<WTmean, 0.85>	79.00(5.77)	78.82(5.21)	77.08(1.86)	77.43(3.24)	76.57(1.75)	76.83(2.50)	77.78(3.80)	77.32(1.81)	77.40(3.96)	78.55(5.57)
<WTmean, 0.95>	313.91(7.80)	315.03(9.65)	317.73(10.23)	314.28(6.73)	315.00(8.99)	314.27(10.54)	313.33(6.84)	311.83(5.18)	314.21(9.48)	316.62(11.00)
Average Rank	5.97	5.92	5.68	5.91	5.1	5.11	5.47	5.14	5.33	5.37

Table 6: The mean (standard deviation) of test objective values of manually rules, GP and tabuGP over 30 independent runs in nine scenarios.

Scenarios	Manually Rules	GP	tabuGP
<Fmean, 0.75>	436.46	336.40(1.84)(↑)	335.40(0.73)(↑)(↑)
<Fmean, 0.85>	502.30	386.56(3.92)(↑)	384.64(1.64)(↑)(↑)
<Fmean, 0.95>	763.85	557.32(9.89)(↑)	552.98(6.26)(↑)(≈)
<Tmean, 0.75>	56.31	13.46(0.75)(↑)	13.30(0.65)(↑)(≈)
<Tmean, 0.85>	103.96	40.31(1.79)(↑)	39.31(0.72)(↑)(↑)
<Tmean, 0.95>	311.71	176.49(4.49)(↑)	178.11(5.38)(↑)(≈)
<WTmean, 0.75>	121.52	26.89(0.77)(↑)	26.48(0.39)(↑)(↑)
<WTmean, 0.85>	221.45	75.63(3.09)(↑)	74.58(1.56)(↑)(↑)
<WTmean, 0.95>	613.94	298.25(14.32)(↑)	294.66(8.26)(↑)(≈)

better than, worse than, or similar to its counterpart. This paper focuses on minimisation problems, where a smaller value denotes better performance.

5.1 Sensitivity Analyses of Parameter *tries* for tabuGP

As introduced in Section 3, the parameter *tries* reflects the efforts to find different offspring. Compared with a smaller value of *tries*, a larger *tries* suggests a high probability to get unique offspring to explore unseen areas during the evolutionary process of GP. However, a larger *tries* value may also indicate a possible long time to find unseen offspring. It is not clear which is a good value for *tries* of tabuGP in DFJSS.

To investigate a good value for *tries*, this subsection does parameter sensitivity analyses for *tries*. Specifically, this paper gives *tries* 10 values from 5 to 50 with a step of 5. For a *tries* value of 5, this indicates that the proposed algorithm will try 5 times to generate unseen offspring with the same pair of parents. Table 5 shows the fitness represented by the mean and standard deviation obtained on training instance of tabuGP with different numbers of *tries* according to 30 independent runs in nine scenarios. The algorithm tabuGP_{*i*} represents the algorithm tabuGP with *i* tries. From the average rank, we can see that tabuGP with a *tries* of 25 achieves the best performance with the smallest rank of 5.1 on the training instances. The performance of tabuGP with smaller *tries* than 25 (ranked as 5.97, 5.92, 5.68, and 5.91) is worse than the ones with a larger *tries* than 25 (ranked as 5.11, 5.47, 5.14, 5.33 and 5.37). The observed pattern is consistent with our intuition that a smaller *tries* value does not achieve enough unseen individuals. With a large *tries*, it might be still not possible to find unseen individuals and ends up with keeping seen individuals, which is not helpful to further improve the performance of the tabuGP algorithm. Overall, we can see that *tries* with 25 is a good starting point to use. Thus, this paper chooses to use *tries* of 25 for further experiment studies, and the corresponding proposed algorithm is represented by tabuGP for simplicity in the later contents.

5.2 Quality of Evolved Scheduling Heuristics

5.2.1 Statistical Test

In nine scenarios, Table 6 shows manually rules, GP and tabuGP' test objective values represented by the mean and standard deviation according to 30 independent runs. Clearly, the learned scheduling rules by GP and tabuGP are significantly better than manually designed rules in all scenarios. The results also shows that tabuGP is significantly better than GP in 5 out of 9 scenarios, and tabuGP achieves comparable performance with GP in other scenarios. In addition, the mean values and standard

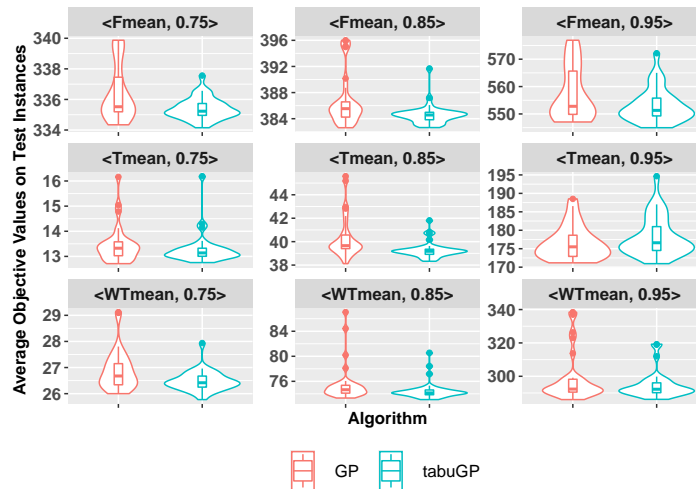


Figure 6: Violin plots of the objective values on test instances of GP and tabuGP based on 30 independent runs in nine scenarios.

deviations of tabuGP are smaller than GP in 8 out of 9 scenarios, which are highlighted in bold. This verifies the effectiveness of the proposed tabuGP algorithm. This also indicates that the proposed strategy of using tabu lists to help explore unseen areas of GP is effective in learning scheduling heuristics for DFJSS.

5.2.2 Violin Plots for the Learned Best Scheduling Heuristics

To delve deeper into the objectives achieved by the algorithms, Figure 6 shows the violin plot of the average objective values on test instances of GP and tabuGP in nine scenarios according to 30 independent runs. In terms of the value distributions, it is evident that the proposed tabuGP algorithm demonstrates its superiority by yielding smaller objective values across the majority of examined scenarios. More importantly, the violin plots of tabuGP are flatter than GP with smaller standard deviations. This shows the good stability of the proposed tabuGP to learn effective scheduling heuristics.

5.2.3 Curves of Average Objective Values of Learned Scheduling Heuristics on Test Instances

Figure 7 shows the changes of GP and tabuGP' average objective values on test instances generation by generation in nine scenarios according to 30 independent runs. The results suggest that tabuGP has the capability to surpass GP in learning scheduling heuristics from the initial stages and sustain this advantage throughout the entire evolutionary process. This further verifies the effectiveness of the proposed tabuGP with developed tabu lists by guiding the search to unexplored areas of GP.

5.3 Training versus Test

Figure 8 shows the scatter plots of the objective values on test instances and fitness on training instances of the best individuals for GP and tabuGP according to 30 independent runs in nine scenarios. First, we can see that there is a high correlation between the training performance and test performance obtained by GP and tabuGP. This indicates a good generalisation ability of the evolved scheduling heuristics by GP algorithms. Second, the results show that the scatter points of tabuGP distribute more at the left

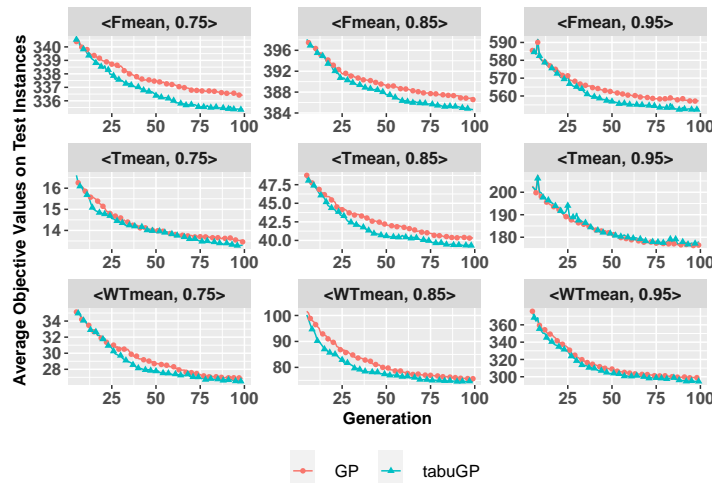


Figure 7: Curves of the average objective values of GP and tabuGP on test instances derived from 30 independent runs across nine scenarios.

bottom corners, which shows the effectiveness of the proposed tabuGP algorithm to get smaller objective values.



Figure 8: Scatter plots of the objective values on test instances and fitness on training instances of the best learned individuals for GP and tabuGP according to 30 independent runs in nine scenarios.

5.4 Diversity of Algorithms

We use entropy to measure the diversity of GP population for DFJSS by using phenotypic characterisations of GP individuals. The entropy is calculated as $entropy = -\sum_{c \in C} \frac{|c|}{|individuals|} \log(\frac{|c|}{|individuals|})$, where C is the set of clusters obtained by the DB-

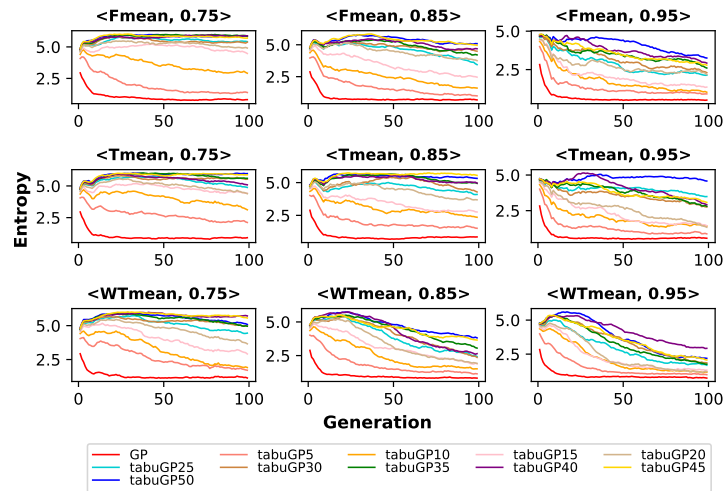


Figure 9: Curves of diversity represented as entropy values of GP and tabuGP with different numbers of *tries* over generations according to 30 independent runs in nine scenarios.

Scan clustering algorithm (Ester et al., 1996; Ardeh et al., 2021) with the phenotypic distance measure (Hildebrandt and Branke, 2015) and a cluster radius of zero. A greater entropy value signifies increased diversity within the population.

Figure 9 shows the curves of diversity represented as entropy values of GP and tabuGP with different numbers of *tries* over generations based on 30 independent runs in nine scenarios. The figure shows that GP loses its population diversity quickly and stays at a low diversity after about generation 5. This may limit the search effectiveness of GP, which is a drawback of traditional GP on DFJSS. Overall, as the number of *tries* increases, the diversities of the corresponding tabuGP algorithms are improved. For example, the diversity of tabuGP10 is clearly larger than tabuGP5, and the diversity of tabuGP15 is obviously larger than tabuGP10 in most scenarios. We also observe that if the *tries* value is larger than 25, a further increase of the *tries* value can increase the algorithm diversity, however, the increment is marginal. This indicates a *tries* value of 25 is a good starting point. This finding is consistent with our observation in Section 5.1.

5.5 Training Time

For GP in DFJSS, the most time-consuming part is the fitness evaluations of individuals. Thus, $O(\text{time}) = \text{evals} * O(\text{eval}) = \text{evals} * \text{ops} * O(\text{ds}) = \text{evals} * \text{ops} * O((M + 100) * \text{rulesize}) = \text{evals} * \text{ops} * O(\text{rulesize})$, where evals, ops, and ds are the number of evaluations, operations, and decision situations, respectively. M (i.e., 10 in this paper) is the number of machines, and 100 is the maximal number of operations in a queue of a machine (the simulation will terminate if the queue size exceeds 100). Each operation will have routing decisions to be allocated to a machine, and sequencing decisions to be selected for processing next. *rulesize* is the number of nodes of a GP individual.

Table 7 shows the mean and standard deviation of the training time of GP and tabuGP according to 30 independent runs in nine scenarios. The results indicate that while tabuGP shows a slight increase in time compared to GP, the difference is minimal. In other words, using the built tabu list and searching with the tabu list are efficient,

Table 7: The mean (standard deviation) of training time of GP and tabuGP over 30 independent runs in nine scenarios.

Scenarios	GP	tabuGP
<Fmean, 0.75>	115(21)	132(28)
<Fmean, 0.85>	120(20)	136(25)
<Fmean, 0.95>	119(14)	147(24)
<Tmean, 0.75>	100(19)	113(23)
<Tmean, 0.85>	113(22)	141(27)
<Tmean, 0.95>	118(17)	156(26)
<WTmean, 0.75>	110(28)	129(22)
<WTmean, 0.85>	108(20)	152(37)
<WTmean, 0.95>	126(26)	163(28)

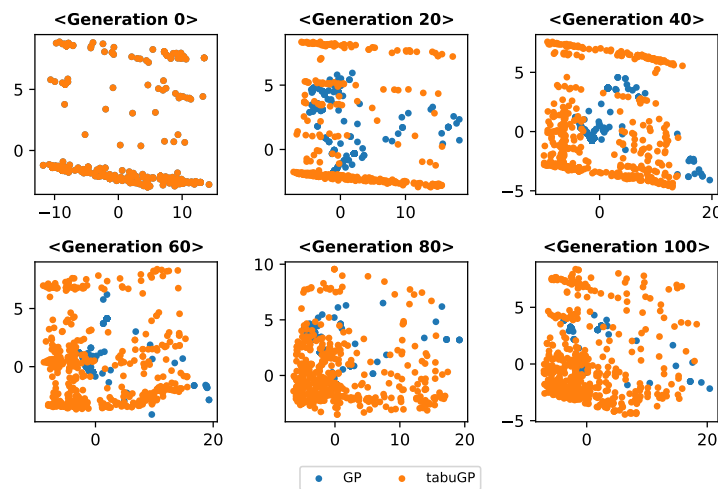


Figure 10: Distribution visualisations for the population individuals of GP and tabuGP according to the phenotypic characterisations with principal component analysis (PCA) at generation 0, 20, 40, 60, 80, and 100.

and do not bring extra computational cost. This shows the efficiency of the proposed tabuGP algorithm, and the efficiency the way of implementing tabu list as introduced in Section 3.2.

6 Further Analysis

6.1 Distributions of Explored Areas

As mentioned earlier, the proposed tabuGP tries to explore more unseen areas during the evolutionary process. This indicates that the individuals of tabuGP are more likely to cover a larger area than GP. This section uses the behaviour of individuals in the population which is represented by phenotypic characterisation to show the exploration areas of GP and tabuGP. Figure 10 shows the distribution visualisations of population individuals of GP and tabuGP according to the phenotypic characterisations with principal component analysis (PCA) at generation 0, 20, 40, 60, 80, and 100. We can see that at the first generation, the behaviour of all the initialised individuals of GP and tabuGP are the same. Along with generations, it is clear that the explored areas by tabuGP are larger than GP. At the later stage of the evolutionary process, the

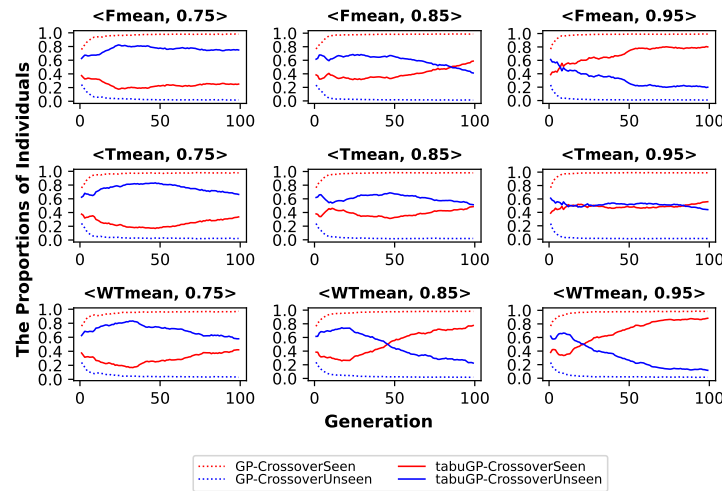


Figure 11: Curves of the proportions of seen individuals and unseen individuals *generated by crossover* of GP and tabuGP along with generations in nine different scenarios, where GP-CrossoverSeen and GP-CrossoverUnseen represents the seen and unseen individuals found by GP, and tabuGP-CrossoverSeen and tabuGP-CrossoverUnseen represents the seen and unseen individuals obtained by tabuGP.

explored areas of tabuGP are still larger than GP, but we can also see most individuals of tabuGP are converged to some areas, i.e., more likely to be the promising areas. This indicated that the proposed tabuGP has a good exploration and exploitation ability.

6.2 The Number of Seen and Unseen Individuals

This section investigates the number of individuals which are seen and unseen in the population. Specifically, the individuals generated by crossover and mutation are examined.

6.2.1 Individuals Generated by Crossover

Figure 11 shows the curves of proportions of seen and unseen individuals found by GP and obtained by tabuGP via crossover in nine scenarios along with generations according to 30 independent runs. We use GP-CrossoverSeen and GP-CrossoverUnseen to represent the seen and unseen individuals found by GP, and tabuGP-CrossoverSeen and tabuGP-CrossoverUnseen to represent the seen and unseen individuals obtained by tabuGP. Comparing with GP-CrossoverSeen and GP-CrossoverUnseen, the results show that the number of unseen individuals (i.e., less than 0.1) is smaller than the number of seen individuals (i.e., larger than 0.9). This means that most of the generated offspring by crossover of GP have been seen in the previous generations or at the current generation. For tabuGP, compared with GP-CrossoverSeen, the number of seen individuals represented by tabuGP-CrossoverSeen is smaller. In addition, the number of unseen individuals of tabuGP represented by tabuGP-CrossoverUnseen is larger than the number of unseen individuals of GP which is represented by GP-CrossoverUnseen. This is the reason why tabuGP can achieve better performance than GP.

For the number of unseen individuals of tabuGP represented by tabuGP-CrossoverUnseen, and the number of seen individuals of tabuGP represented by tabuGP-CrossoverSeen, the results show that the number of unseen individuals is

larger than the number of seen individuals in most scenarios, i.e., $\langle F_{\text{mean}}, 0.75 \rangle$, $\langle F_{\text{mean}}, 0.85 \rangle$, $\langle T_{\text{mean}}, 0.75 \rangle$, $\langle T_{\text{mean}}, 0.85 \rangle$ and $\langle WT_{\text{mean}}, 0.75 \rangle$. However, the number of unseen individuals is larger than the number of seen individuals at the early stages, but vice in verse in the later stages of the evolutionary process of tabuGP in three scenarios, i.e., $\langle F_{\text{mean}}, 0.95 \rangle$, $\langle WT_{\text{mean}}, 0.85 \rangle$ and $\langle WT_{\text{mean}}, 0.95 \rangle$. There is no big difference between the number of unseen and seen individuals of tabuGP in scenario $\langle T_{\text{mean}}, 0.95 \rangle$.

In terms of the trends of curves, Figure 11 shows the number of seen (unseen) individuals of GP generated by crossover increases (decreases) along with generations to a steady state. In addition, there are no obvious pattern changes in the number of seen and unseen individuals for GP. However, it is clear that the number of seen (unseen) individuals of tabuGP increases (decreases) along with generations in all scenarios. The changes in proportions of offspring generated by crossover rely on the utilisation levels. We can see that the curves of tabuGP in the scenarios with the same utilisation level are similar, e.g., in $\langle F_{\text{mean}}, 0.75 \rangle$, $\langle T_{\text{mean}}, 0.75 \rangle$ and $\langle WT_{\text{mean}}, 0.75 \rangle$, and the number of unseen individuals has slightly decrease trends while the number of seen individuals has increase trends. This is clearer in scenarios with utilisation level of 0.85, i.e., $\langle F_{\text{mean}}, 0.85 \rangle$, $\langle T_{\text{mean}}, 0.85 \rangle$ and $\langle WT_{\text{mean}}, 0.85 \rangle$, where the number of seen individuals is equal or larger than the number of unseen at the later stage of the evolutionary process of tabuGP. In the scenarios with utilisation levels of 0.95, the number of seen individuals quickly becomes larger than the number of unseen individuals in scenario $\langle F_{\text{mean}}, 0.95 \rangle$ and $\langle WT_{\text{mean}}, 0.95 \rangle$. It seems that it becomes more difficult to generate unseen individuals when the DFJSS scenarios become harder to address, i.e., with higher utilisation levels. A possible reason is that the individuals needed for complex DFJSS problems have a higher requirement design on particular extreme cases that are more likely to be similar.

6.2.2 Individuals Generated by Mutation

Figure 12 shows the proportions of seen and unseen individuals generated by mutation according to 30 independent runs. For GP, the number of unseen individuals (smaller than 0.2) obtained by mutation is much smaller than the seen individuals (larger than 0.8). On the contrary, the number of unseen individuals (about 0.8) by mutation of tabuGP is much larger than the seen individuals (about or larger than 0.2) in all scenarios. These findings of unseen and seen individuals by mutation are consistent with the observations by crossover.

In terms of unseen and seen individuals obtained by tabuGP, the patterns of curves are also utilisation level dependent. With the increase of utilisation level, although the number of unseen individuals is larger than seen individuals, there is a clear reduction of the number of unseen individuals and an increase of the number of seen individuals in the population of tabuGP. This observation is consistent with the findings from crossover, which indicates it is difficult to generate unseen individuals for busier job shops with larger utilisation levels.

6.3 Alternative Way to Keep Offspring

The key idea of this paper is to increase the search ability of GP for DFJSS by guiding the search to explore more unexplored areas with tabu list. This idea is implemented by controlling the newly generated offspring via crossover and mutation. When generating two offspring by crossover with tabu list, after we generate $Offspring_1$ and $Offspring_2$ from two parents, this paper keeps the two generated individuals originate

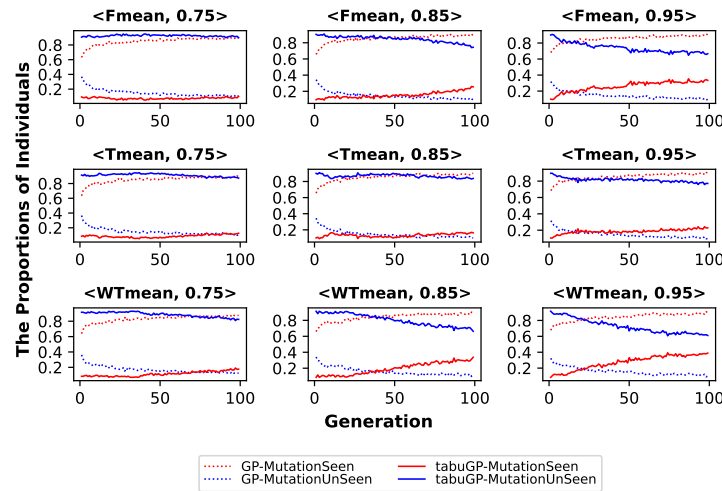


Figure 12: Curves of the proportions of seen individuals and unseen individuals generated by mutation of GP and tabuGP along with generations in nine different scenarios, where GP-MutationSeen and GP-MutationUnSeen represents the seen and unseen individuals found by GP, and tabuGP-MutationSeen and tabuGP-MutationUnSeen represents the the seen and unseen individuals obtained by tabuGP.

from the same two parents. There is another possible way to decide which unseen offspring to use as individuals in the next generation. That is to always save the unseen offspring, either $Offspring_1$ or $Offspring_2$. For example, if $Offspring_1$ is unseen and $Offspring_2$ is seen, we will save $Offspring_1$. If $Offspring_1$ is seen and $Offspring_2$ is unseen, we will save $Offspring_2$. When we get two unseen offspring, we will stop trying, and keep the two unseen offspring in the next generation. If we consider the origin of the generated offspring, they can be either originated from $\langle Parent_1, Parent_1 \rangle$ or $\langle Parent_2, Parent_2 \rangle$. Since this way only considers the quantity of generated offspring but not their origin, we name the algorithm as *tabuGP_quantityOnly*. The unseen offspring of this paper are originated from $\langle Parent_1, Parent_2 \rangle$ as introduced in Section 3.3.

6.3.1 Quality of Learned Scheduling Heuristics

Table 8 shows the mean and standard deviations of test objectives of GP, tabuGP and *tabuGP_quantityOnly* based on 30 independent runs in nine scenarios. The results show that tabuGP achieves the best performance among all compared algorithms with a rank of 1.86. Specifically, tabuGP performs significantly better than GP in five out of nine scenarios, while *tabuGP_quantityOnly* is only significantly better than GP in two out of nine scenarios. This indicates that when generating offspring for the next generation, we should pay attention to the origin of individuals.

6.3.2 Effect on Diversity of TabuGP

To understand the reasons for the performance difference of tabuGP and *tabuGP_quantityOnly*, this section investigates their population diversities. Figure 13 shows the diversity of GP, tabuGP and *tabuGP_quantityOnly* along with generations according to 30 independent runs in nine scenarios. Both tabuGP and *tabuGP_quantityOnly* have a higher diversity than GP, and the *tabuGP_quantityOnly*

Table 8: The mean (standard deviation) of test objective values of GP, *tabuGP* and *tabuGP_quantityOnly* over 30 independent runs in nine scenarios.

Scenarios	GP	tabuGP	tabuGP_quantityOnly
<Fmean, 0.75>	336.40(1.84)	335.40(0.73)(↑)	335.96(1.56)(≈)
<Fmean, 0.85>	386.56(3.92)	384.64(1.64)(↑)	384.73(1.23)(↑)
<Fmean, 0.95>	557.32(9.89)	552.98(6.26)(≈)	551.89(4.85)(≈)
<Tmean, 0.75>	13.46(0.75)	13.30(0.65)(≈)	13.31(0.47)(≈)
<Tmean, 0.85>	40.31(1.79)	39.31(0.72)(↑)	39.44(0.78)(↑)
<Tmean, 0.95>	176.49(4.49)	178.11(5.38)(≈)	175.76(3.74)(≈)
<WTmean, 0.75>	26.89(0.77)	26.48(0.39)(↑)	27.31(1.94)(≈)
<WTmean, 0.85>	75.63(3.09)	74.58(1.56)(↑)	74.49(1.00)(≈)
<WTmean, 0.95>	298.25(14.32)	294.66(8.26)(≈)	295.46(8.34)(≈)
Average Rank	2.17	1.86	1.97

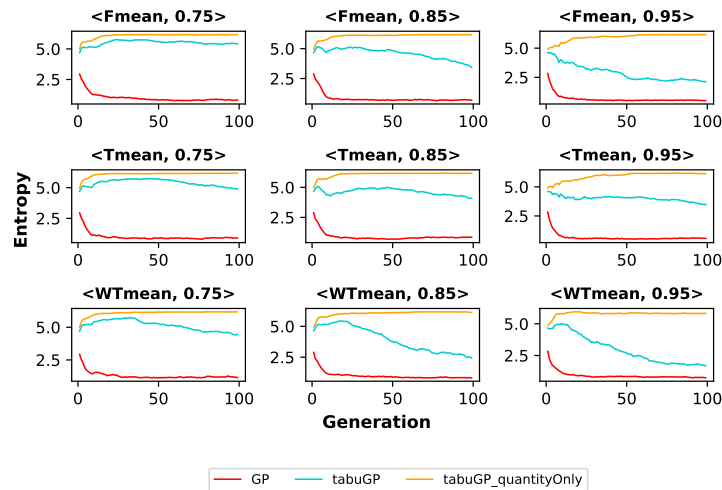


Figure 13: The curves of diversity represented as entropy values of GP, *tabuGP* and *tabuGP_quantityOnly* over generations according to 30 independent runs in nine scenarios.

keeps a higher diversity along with the generations. *tabuGP* maintains a higher diversity but has a clear diversity decrease along with generations, which is a sign that the proposed *tabuGP* can get a good balance of exploration and exploitation. This is consistent with our findings for the distributions of explored areas of *tabuGP* in Section 6.1. Further looking at the curves of *tabuGP*, we can see the trends of the curves are similar to the *tabuGP-CrossoverUnseen* and *tabuGP-MutationUnseen* in Figure 11 and Figure 12. This indicates that population diversity is affected by the proposed strategies to keep unseen individuals with built *tabu* lists.

6.4 Insights of Learned Scheduling Heuristics

Figure 14 shows one of the best learned sequencing rule obtained by *tabuGP* in scenario <WTmean, 0.95>. We can see that W is the most important feature for this sequencing rule (appears 7 times), followed by feature WKR (appears 4 times). PT is also important for this sequencing rule to order the processing of operations in a queue of a machine (appears 3 times). In addition, we find that PT/W plays an important role as a learned building block for this sequencing rule, which is a commonly used rule, i.e., weighted

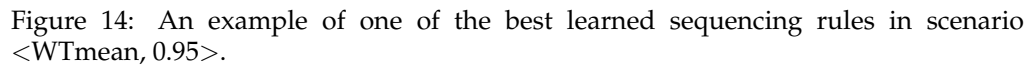
[illegible]

Figure 15 shows the paired routing rule of the sequencing rule as shown in Figure 14. The top three important features are MWT, WIQ and NIQ, which appear 7, 5, and 5 times, respectively. We can also see W^*MWT and $OWT/(WIQ^*MWT)$ are two learned building blocks which are important for this routing rule, which appear 3 and 2 times. These findings are consistent with our preliminary work about feature importance of learned scheduling rules in DFJSS (Zhang et al., 2020a, 2021b).

The goal of this paper is to propose an effective GP algorithm with tabu lists for DFJSS. The goal has been successfully achieved by the proposed strategies to guide the search

of GP to explore more unexplored areas. The idea is realised by controlling the generation of unseen individuals with built tabu lists of visited individuals.

The results show that the proposed tabuGP can achieve effective scheduling heuristics in various DFJSS scenarios. Analyses on the parameter of controlling the pressure of bringing unseen individuals to the evolutionary process of GP show that there is a marginal value for exploring unseen individuals, further increasing the value does not help too much. The analyses on the number of unseen individuals and population diversity during the evolutionary process indicate that the effectiveness of the proposed algorithm benefits from the diversity brought by unseen individuals and the origins of unseen individuals. This paper also visualises the population individuals by using their phenotypic characterisations and PCA technique. The results show that the proposed algorithm does cover a larger exploration areas. Meanwhile, the proposed algorithm also has the ability to explore promising areas. Details of other alternative way to keep offspring to the next generation are also discussed to give readers more information on implementing the proposed algorithm to specific problems.

We will investigate some interesting directions in future. We plan to design an interactive GP approach to let GP explore more in the search space that is given as human preferences. In addition, a more advanced strategy to explore unseen individuals of tabuGP will be investigated. Theoretical studies of GP on DFJSS will be studied.

References

- Ardeh, M. A., Mei, Y., and Zhang, M. (2021). A novel multi-task genetic programming approach to uncertain capacitated arc routing problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 759–767.
- Biegel, J. E. and Davern, J. J. (1990). Genetic algorithms and job shop scheduling. *Computers & Industrial Engineering*, 19(1-4):81–91.
- Braune, R., Benda, F., Doerner, K. F., and Hartl, R. F. (2022). A genetic programming learning approach to generate dispatching rules for flexible shop scheduling problems. *International Journal of Production Economics*, 243:108342.
- Brucker, P. and Schlie, R. (1990). Job-shop scheduling with multi-purpose machines. *Computing*, 45(4):369–375.
- Bülbül, K. and Kaminsky, P. (2013). A linear programming-based method for job shop scheduling. *Journal of Scheduling*, 16:161–183.
- Burke, E., Gustafson, S., and Kendall, G. (2002). A survey and analysis of diversity measures in genetic programming. In *Proceedings of the Annual Conference on Genetic and Evolutionary Computation*, pages 716–723.
- Burke, E. K., Gustafson, S., and Kendall, G. (2004). Diversity in genetic programming: An analysis of measures and correlation with fitness. *IEEE Transactions on Evolutionary Computation*, 8(1):47–62.
- Canbolat, Y. B. and Gundogar, E. (2004). Fuzzy priority rule for job shop scheduling. *Journal of intelligent manufacturing*, 15:527–533.
- Chen, B. and Matis, T. I. (2013). A flexible dispatching rule for minimizing tardiness in job shop scheduling. *International Journal of Production Economics*, 141(1):360–365.

F. Zhang, M. Ardeh, Y. Mei, and M. Zhang

- Davis, L. (2014). Job shop scheduling with genetic algorithms. In *Proceedings of the International Conference on Genetic Algorithms and their Applications*, pages 136–140. Psychology Press.
- De Arruda Pereira, M., Júnior, C. A. D., Carrano, E. G., and De Vasconcelos, J. A. (2014). A niching genetic programming-based multi-objective algorithm for hybrid data classification. *Neurocomputing*, 133:342–357.
- Destouet, C., Tlahig, H., Bettayeb, B., and Mazari, B. (2023). Flexible job shop scheduling problem under industry 5.0: A survey on human reintegration, environmental consideration and resilience improvement. *Journal of Manufacturing Systems*, 67:155–173.
- D’Haen, R., Braekers, K., and Ramaekers, K. (2023). Integrated scheduling of order picking operations under dynamic order arrivals. *International Journal of Production Research*, 61(10):3205–3226.
- Durasevic, M. and Jakobovic, D. (2018). A survey of dispatching rules for the dynamic unrelated machines environment. *Expert Systems with Applications*, 113:555–569.
- Ester, M., Kriegel, H. P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231.
- Galván, E. and Schoenauer, M. (2019). Promoting semantic diversity in multi-objective genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1021–1029.
- Gere Jr, W. S. (1966). Heuristics in job shop scheduling. *Management Science*, 13(3):167–190.
- Hart, E. and Sim, K. (2016). A hyper-heuristic ensemble method for static job-shop scheduling. *Evolutionary Computation*, 24(4):609–635.
- Hildebrandt, T. and Branke, J. (2015). On using surrogates with genetic programming. *Evolutionary Computation*, 23(3):343–367.
- Hildebrandt, T., Heger, J., and Scholz-Reiter, B. (2010). Towards improved dispatching rules for complex shop floor scenarios: a genetic programming approach. In *Proceedings of the Conference on Genetic and Evolutionary Computation*, pages 257–264. ACM.
- Jackson, D. (2010). Promoting phenotypic diversity in genetic programming. In *International Conference on Parallel Problem Solving from Nature*, pages 472–481. Springer.
- Jaklinović, K., Dhurasević, M., and Jakobović, D. (2021). Designing dispatching rules with genetic programming for the unrelated machines environment with constraints. *Expert Systems with Applications*, 172:114548.
- Juárez-Smith, P., Trujillo, L., García-Valdez, M., Fernández de Vega, F., and Chávez, F. (2019). Local search in speciation-based bloat control for genetic programming. *Genetic Programming and Evolvable Machines*, 20:351–384.
- Kaban, A., Othman, Z., and Rohmah, D. (2012). Comparison of dispatching rules in job-shop scheduling problem using simulation: a case study. *International Journal of Simulation Modelling*, 11(3):129–140.

- Kelly, J., Hemberg, E., and O'Reilly, U.-M. (2019). Improving genetic programming with novel exploration-exploitation control. In *European Conference on Genetic Programming*, pages 64–80. Springer.
- Koza, J. R. and Poli, R. (2005). Genetic programming. In *Search Methodologies*, pages 127–164. Springer.
- Langdon, W. B. and Poli, R. (2013). *Foundations of genetic programming*. Springer Science & Business Media.
- Li, X. and Gao, L. (2016). An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. *International Journal of Production Economics*, 174:93–110.
- Liu, A., Luh, P. B., Yan, B., and Bragin, M. A. (2021). A novel integer linear programming formulation for job-shop scheduling problems. *IEEE Robotics and Automation Letters*, 6(3):5937–5944.
- Maier, T., Sanders, P., and Dementiev, R. (2019). Concurrent hash tables: Fast and general (!) *ACM Transactions on Parallel Computing*, 5(4):1–32.
- Manne, A. S. (1960). On the job-shop scheduling problem. *Operations Research*, 8(2):219–223.
- Nguyen, S., Mei, Y., and Zhang, M. (2017). Genetic programming for production scheduling: a survey with a unified framework. *Complex & Intelligent Systems*, 3(1):41–66.
- Nguyen, S., Zhang, M., Alahakoon, D., and Tan, K. C. (2018). Visualizing the evolution of computer programs for genetic programming. *IEEE Computational Intelligence Magazine*, 13(4):77–94.
- Nguyen, S., Zhang, M., Johnston, M., and Tan, K. C. (2015). Automatic programming via iterated local search for dynamic job shop scheduling. *IEEE Transactions on Cybernetics*, 45(1):1–14.
- Nicolau, M. and Fenton, M. (2016). Managing repetition in grammar-based genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 765–772.
- Ono, K., Hanada, Y., Kumano, M., and Kimura, M. (2019). Enhancing island model genetic programming by controlling frequent trees. *Journal of Artificial Intelligence and Soft Computing Research*, 9(1):51–65.
- Prajapati, V. K., Jain, M., and Chouhan, L. (2020). Tabu search algorithm (tsa): A comprehensive survey. In *Proceedings of International Conference on Emerging Technologies in Computer Engineering: Machine Learning and Internet of Things*, pages 1–8. IEEE.
- Salama, S., Kaihara, T., Fujii, N., and Kokuryo, D. (2022). Multi-objective approach with a distance metric in genetic programming for job shop scheduling. *International Journal of Automation Technology*, 16(3):296–308.
- Sitahong, A., Yuan, Y., Li, M., Ma, J., Ba, Z., and Lu, Y. (2022). Designing dispatching rules via novel genetic programming with feature selection in dynamic job-shop scheduling. *Processes*, 11(1):65.

F. Zhang, M. Ardeh, Y. Mei, and M. Zhang

- Tay, J. C. and Ho, N. B. (2008). Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. *Computers & Industrial Engineering*, 54(3):453–473.
- Vanneschi, L., Castelli, M., and Silva, S. (2014). A survey of semantic methods in genetic programming. *Genetic Programming and Evolvable Machines*, 15:195–214.
- Xu, M., Mei, Y., Zhang, F., and Zhang, M. (2022). Genetic programming with diverse partner selection for dynamic flexible job shop scheduling. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 615–618.
- Xu, M., Mei, Y., Zhang, F., and Zhang, M. (2023a). Genetic programming for dynamic flexible job shop scheduling: Evolution with single individuals and ensembles. *IEEE Transactions on Evolutionary Computation*, DOI: 10.1109/TEVC.2023.3334626.
- Xu, M., Mei, Y., Zhang, F., and Zhang, M. (2023b). Genetic programming with lexicase selection for large-scale dynamic flexible job shop scheduling. *IEEE Transactions on Evolutionary Computation*, DOI: 10.1109/TEVC.2023.3244607.
- Xu, M., Zhang, F., Mei, Y., and Zhang, M. (2021). Genetic programming with archive for dynamic flexible job shop scheduling. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 2117–2124. IEEE.
- Zhang, F., Mei, Y., Nguyen, S., Tan, K. C., and Zhang, M. (2022a). Instance rotation based surrogate in genetic programming with brood recombination for dynamic job shop scheduling. *IEEE Transactions on Evolutionary Computation*, 27(5):1192–1206.
- Zhang, F., Mei, Y., Nguyen, S., Tan, K. C., and Zhang, M. (2022b). Multitask genetic programming based generative hyper-heuristics: A case study in dynamic scheduling. *IEEE Transactions on Cybernetics*, 52(10):10515–10528.
- Zhang, F., Mei, Y., Nguyen, S., and Zhang, M. (2020a). 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*, pages 214–230. Springer.
- Zhang, F., Mei, Y., Nguyen, S., and Zhang, M. (2020b). Guided subtree selection for genetic operators in genetic programming for dynamic flexible job shop scheduling. In *Proceedings of the European Conference on Genetic Programming*, pages 262–278. Springer.
- Zhang, F., Mei, Y., Nguyen, S., and Zhang, M. (2020c). A preliminary approach to evolutionary multitasking for dynamic flexible job shop scheduling via genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 107–108. ACM.
- Zhang, F., Mei, Y., Nguyen, S., and Zhang, M. (2021a). Correlation coefficient based recombinative guidance for genetic programming hyper-heuristics in dynamic flexible job shop scheduling. *IEEE Transactions on Evolutionary Computation*, 25(3).
- Zhang, F., Mei, Y., Nguyen, S., and Zhang, M. (2021b). Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job shop scheduling. *IEEE Transactions on Cybernetics*, 51(4):1797–1811.

- Zhang, F., Mei, Y., Nguyen, S., and Zhang, M. (2022c). Learning strategies on scheduling heuristics of genetic programming in dynamic flexible job shop scheduling. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE.
- Zhang, F., Mei, Y., Nguyen, S., and Zhang, M. (2022d). Multitask multiobjective genetic programming for automated scheduling heuristic learning in dynamic flexible job-shop scheduling. *IEEE Transactions on Cybernetics*.
- Zhang, F., Mei, Y., Nguyen, S., and Zhang, M. (2022e). Phenotype based surrogate-assisted multi-objective genetic programming with brood recombination for dynamic flexible job shop scheduling. In *Proceedings of the IEEE Symposium Series on Computational Intelligence*, pages 1218–1225. IEEE.
- Zhang, F., Mei, Y., Nguyen, S., and Zhang, M. (2023a). Survey on genetic programming and machine learning techniques for heuristic design in job shop scheduling. *IEEE Transactions on Evolutionary Computation*, 28(1):147–167.
- Zhang, F., Mei, Y., Nguyen, S., Zhang, M., and Tan, K. C. (2021c). Surrogate-assisted evolutionary multitask genetic programming for dynamic flexible job shop scheduling. *IEEE Transactions on Evolutionary Computation*, 25(4):651–665.
- Zhang, F., Mei, Y., and Zhang, M. (2018). Genetic programming with multi-tree representation for dynamic flexible job shop scheduling. In *Proceedings of the Australasian Joint Conference on Artificial Intelligence*, pages 472–484. Springer.
- Zhang, F., Mei, Y., and Zhang, M. (2019). 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*, pages 33–49. Springer.
- Zhang, F., Mei, Y., and Zhang, M. (2023b). An investigation of terminal settings on multitask multi-objective dynamic flexible job shop scheduling with genetic programming. In *Proceedings of the Conference on Genetic and Evolutionary Computation*, pages 259–262.
- Zhou, Y., Yang, J., and Huang, Z. (2020). Automatic design of scheduling policies for dynamic flexible job shop scheduling via surrogate-assisted cooperative co-evolution genetic programming. *International Journal of Production Research*, 58(9):2561–2580.
- Zhu, L., Zhang, F., Zhu, X., Chen, K., and Zhang, M. (2023). Sample-aware surrogate-assisted genetic programming for scheduling heuristics learning in dynamic flexible job shop scheduling. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 384–392.