Genetic Programming for Dynamic Flexible Job Shop Scheduling: Evolution with Single Individuals and Ensembles

Meng Xu, *Student Member, IEEE,* Yi Mei, *Senior Member, IEEE,*
Fangfang Zhang, *Member, IEEE,* and Mengjie Zhang, *Fellow, IEEE*

***Abstract*—Dynamic flexible job shop scheduling is an important but difficult combinatorial optimisation problem that has numerous real-world applications. Genetic programming has been widely used to evolve scheduling heuristics to solve this problem. Ensemble methods have shown promising performance in many machine learning tasks, but previous attempts to combine genetic programming with ensemble techniques are still limited and require further exploration. This paper proposes a novel ensemble genetic programming method that uses a population consisting of both single individuals and ensembles. The main contributions include: 1) developing a genetic programming method that evolves a population comprising both single individuals and ensembles, allowing breeding between them to explore the search space more effectively; 2) proposing an ensemble construction and selection strategy to form ensembles by selecting diverse and complementary individuals; and 3) designing new crossover and mutation operators to produce offspring from single individuals and ensembles. Experimental results demonstrate that the proposed method outperforms existing traditional and ensemble genetic programming methods in most scenarios. Further analyses find that the success is attributed to the enhanced population diversity and extensive search space exploration achieved by the proposed method.**

***Index Terms*—heuristic learning, genetic programming, ensemble, dynamic scheduling.**

I. Introduction

DYNAMIC flexible job shop scheduling (DFJSS) stands as a significant combinatorial optimisation challenge that has received extensive interest from academia and industry [1]–[5]. For DFJSS, two types of decision points need to be considered: the routing and sequencing decision points [6]–[9]. A routing decision point is when an operation is ready and a machine needs to be selected from a number of candidate machines. A sequencing decision point is when a machine is available and an operation needs to be selected from its waiting queue. To tackle the DFJSS problem, scheduling heuristics have been widely used, typically comprising a routing rule and a sequencing rule [10]–[12]. These rules are priority functions used to assign priorities to available machines/operations when meeting routing or sequencing decision points. Nonetheless, the manual construction of such heuristics proves to be labor-intensive and requires significant domain knowledge [13].

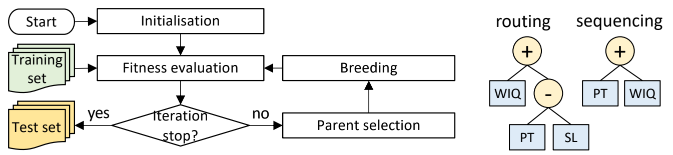
Meng Xu, Yi Mei, Fangfang Zhang, and Mengjie Zhang are with the Centre for Data Science and Artificial Intelligence & School of Engineering and Computer Science, Victoria University of Wellington, Wellington 6140, New Zealand (e-mail: meng.xu@ecs.vuw.ac.nz; yi.mei@ecs.vuw.ac.nz; fangfang.zhang@ecs.vuw.ac.nz; mengjie.zhang@ecs.vuw.ac.nz).

Genetic programming (GP) has been successfully used to automatically evolve effective scheduling heuristics for DFJSS [14]–[18]. Most of the existing GP methods for DFJSS mainly focus on evolving one scheduling heuristic [19], [20]. Recently, there is a growing trend to learn a group of scheduling heuristics and leverage this group to make decisions, allowing for further exploration of the search space and the discovery of high-quality solutions [21]. Ensemble GP (EGP) is a variant of GP, incorporating GP with ensemble learning, which is able to combine multiple scheduling heuristics into an ensemble to make a decision [22]. In EGP, each scheduling heuristic in an ensemble is expected to be different and complementary to others [23]. The output of each scheduling heuristic in an ensemble is then combined using an aggregation function, such as voting or averaging, to make a final decision [24]. The goal of EGP is to improve the performance and generalisation ability of GP by leveraging the strengths of multiple scheduling heuristics while mitigating their weaknesses [25]. In this way, the probability of making a sub-optimal decision in a given decision point is reduced, as the elements can complement each other to avoid poor decisions. Consequently, an ensemble is expected to be more stable than a single scheduling heuristic, which can also provide confidence and trust for users, especially in real-world applications [26]. EGP has been applied in various domains, such as regression [27] and classification [24], [28], [29], and has shown promising results in terms of performance and generalisation ability.

Recently, EGP has been applied to evolve a group of scheduling heuristics for solving scheduling problems [30]–[33]. The existing studies about EGP for scheduling problems are still in their early stage and can be divided into three categories. The first category uses multiple subpopulations to evolve scheduling heuristics simultaneously, then the best scheduling heuristics from each subpopulation are grouped to form an ensemble [34]. For this type of method, each scheduling heuristic obtained is likely to be of high-quality. However, they usually do not consider the cooperation between them, as the evolution of each scheduling heuristic is carried out independently. Alternatively, they just consider the cooperation of the scheduling heuristics when performing the evaluation, not during the evolution. The second category uses a single population to evolve scheduling heuristics, then selects a subset of scheduling heuristics from the population in the last generation to form the ensemble [35]. This type of method can follow the conventional evolutionary process, consider population diversity (e.g., by niching), or evolve based on different subsets of training instances to obtain a diverse set of effective scheduling heuristics. After obtaining a group of scheduling heuristics as candidates, this type of method normally uses extra training instances to construct an ensemble with the aid of some greedy selection methods. Although the diversity issue can be considered for this type of method, they still do not consider the cooperation between these scheduling heuristics, and the evolution of each scheduling heuristic still proceeds independently. Moreover, the greedy selection process using extra training instances requires the evaluation on more instances, which increases the computational time. The above two categories do not consider the evolution of ensembles. The third category explores the evolution of both single individuals and ensembles [36]. However, the evolutionary processes of single individuals and ensembles are independent. For example, this type of method typically evolves a set of scheduling heuristics as candidates using GP and then employs meta-heuristic methods (e.g., genetic algorithm) to evolve an ensemble. In such cases, single individuals and ensembles cannot directly contribute to each other. Overall, the existing studies on EGP for solving scheduling problems are still in an early stage and need to be further explored. A novel EGP method is required that has the potential to identify the strengths of both single individuals and ensembles, and make them mutually reinforcing. On the other hand, the lexicase selection is a very effective technique to improve population diversity, having the principle to evolve mutual complementary expert individuals that are good for handling different cases [37]. This highly motivates us to incorporate lexicase selection and ensemble techniques so that diverse individuals can be found by lexicase selection to form an ensemble.

This paper proposes a novel EGP method for solving DFJSS, which contains a new ensemble construction and selection strategy to select diverse individuals to form ensembles and considers the evolution of single individuals and ensembles together within a single population. Our method aims to exploit the strengths of both single individuals and ensembles to effectively solve the DFJSS problem. Specifically, the key contributions of this paper are summarised as follows:

1) This paper proposes a new EGP method for DFJSS, called EGP^e, which enables the **e**volution between single individuals and ensembles, offering more flexibility in the evolutionary process and better exploration of search space. Experimental results demonstrate that EGP^e outperforms the standard GP and existing EGP methods in solving the DFJSS problem.
2) This paper designs a new ensemble construction and selection strategy to help the proposed EGP^e method select diverse and high-quality individuals into an ensemble. The strategy uses lexicase selection to choose diverse individuals good at solving different cases as candidates and then employs a new similarity-checking technique to further enhance diversity. Experimental results verify the effectiveness of the developed ensemble construction and selection strategy.
3) This paper develops new crossover and mutation operators to facilitate the evolution between single individuals and ensembles in EGP^e. The offspring generated by these operators can be either single individuals or ensembles, resulting in more flexible breeding between single individuals and ensembles. Experimental results verify the effectiveness of the designed new genetic operators.
4) Further analyses show that the scheduling heuristics in a promising ensemble perform differently but do have some similarities, which enable them to make the same decisions in most cases and complement each other at particular decision points. Moreover, this paper provides an interesting research direction that performs evolution with single individuals and ensembles together in ensemble learning.



(a) The GP hyper-heuristic for DFJSS (b) Individual

Fig. 1: The flowchart depicting the GP hyper-heuristic method and a representation example of the individual for DFJSS.

II. Background

A. GP for DFJSS

GP is a type of hyper-heuristic method [38] that has been successfully used to learn scheduling heuristics for solving the DFJSS problem [14], [16], [18], [39]. The process involves two important steps: the training process and the test process [2]. During the training process, GP is used along with a set of training instances to evolve a scheduling heuristic [13]. This evolved scheduling heuristic is then applied to unseen test instances to verify its generalisation ability [40].

The flowchart of the GP hyper-heuristic for evolving scheduling heuristics to address the DFJSS problem is depicted in Fig. 1(a). The process starts by randomly initialising a population of individuals, where each individual represents a scheduling heuristic for DFJSS [41]. Each scheduling heuristic is composed of two rules: one for routing and the other for sequencing, both represented by tree structures [20], as shown in Fig. 1(b). WIQ represents work remaining in the queue, PT is the processing time, and SL means the slack. After population initialisation, fitness evaluation is carried out to measure the performance of each individual on the training instance(s) by making decisions on each decision point and finally obtaining the scheduling result. The parent selection process then selects high-quality individuals as parents for breeding offspring for the next generation. The breeding process involves reproduction, crossover, and mutation. This process repeats until the termination condition is met.

B. Related Work

Ensemble learning is a widely used technique in the machine learning community [30], [42]. Ensemble learning [43] involves combining the decisions of multiple models to improve the overall performance of the system. The idea is to leverage the diversity and complementarity of different models to create a stronger and more accurate ensemble model than each element model. EGP is a technique that combines the power of GP with the idea of ensemble learning [36]. Recently, EGP has gained attention in the research community for learning a group of scheduling heuristics [44].

In [45], NELLI-GP was proposed by extending a classical ensemble method called NELLI, to evolve an ensemble of scheduling heuristics for solving the job shop scheduling problem. NELLI-GP adopts a divide-and-conquer strategy in which each scheduling heuristic in the ensemble solves a unique subset of instances. Experiments show that an ensemble of scheduling heuristics can be evolved that outperforms existing scheduling heuristics and recent hyper-heuristic methods. In [34], an ensemble of scheduling heuristics was evolved using cooperative coevolution GP for solving the static job shop scheduling problem and this method can produce more robust scheduling heuristics than the classical GP. The cooperative coevolution method contains the same number of subpopulations as the number of elements in the ensemble, and each subpopulation is used to evolve a single scheduling heuristic for the ensemble. However, these works only consider a static scheduling environment. Further, the individuals (scheduling heuristics) from different subpopulations interact with each other only when they are evaluated together as an ensemble.

Several studies have explored the use of ensemble methods in dynamic scheduling problems. In [30], four ensemble learning methods were investigated to create ensembles of scheduling heuristics for solving the dynamic scheduling problem. The four ensemble learning methods apply simple ensemble combinations, BagGP, BoostGP, and cooperative coevolution GP. The results show that creating ensembles of scheduling heuristics by simple ensemble combination, BagGP, and BoostGP is able to obtain better scheduling results than the standard GP method. In [46], a preliminary investigation about using GP to evolve ensembles of scheduling heuristics for solving the dynamic job shop scheduling problem was done. The grouping is done randomly, and multiple groupings are made for a single individual to find the overall performance when it is part of an ensemble. However, the proposed method does not show a significantly better performance than the standard GP method. Some studies conducted investigations about comparing different aggregation methods for forming the ensemble [25], [47]. In [47], by comparing different aggregation methods, it shows that the voting aggregation method is much more stable than other aggregation methods. In [48], the authors applied EGP and the multilevel GP on solving the dynamic job shop scheduling problem and demonstrated that using ensembles can improve the performance compared to single scheduling heuristics.

In addition to these studies that focus on investigating classical ensemble learning methods, some studies proposed new ideas beyond this. In [49], the authors studied different ensemble learning methods and proposed an ensemble subset selection method to remove the elements from the ensemble that do not contribute to its quality using extra training instances. However, the use of extra training instances requires additional instances and is more time-consuming. In [50], the authors applied the niching mechanism to cooperative coevolution GP to evolve an ensemble for the dynamic job shop scheduling. The fitness is decided by the performance and the diversity belonging to an ensemble by calculating the phenotypic distance. The results show that the niched cooperative coevolution GP method obtains similar performance to the baseline cooperative coevolution GP methods, but has smaller rule sizes. In [51], a new ensemble constructing method was proposed that allows ensembles to consist not only of scheduling heuristics but also of other ensembles. This method can obtain better performance but takes high computational cost.

Some studies combine GP with meta-heuristic algorithms, first learning a set of scheduling heuristics by GP, then learning ensembles by meta-heuristic algorithms [31]–[33]. For example, in [31], a hybrid algorithm by combining GP and genetic algorithm was proposed to evolve an ensemble of scheduling heuristics for solving an online one-machine scheduling problem. This method interleaves GP and genetic algorithm, GP is for generating scheduling heuristics and genetic algorithm is to evolve ensembles from the scheduling heuristics produced by the GP. This method is able to explore different combinations of the ensemble by genetic algorithm. However, the use of genetic algorithm to evolve ensembles requires additional training time, and they only consider the one-machine scheduling problem, which is less complex than the DFJSS problem investigated in this paper.

In summary, the study in solving the scheduling problems with GP and ensemble learning is limited. Most of the studies are mainly about the investigation or comparison of classical ensemble methods applied to scheduling problems, which cannot provide significantly better performance but suggests that the voting aggregation method can support better stability than other aggregation methods. Moreover, the existing studies about combining GP with ensemble technique to scheduling problems mainly focus on evolving scheduling heuristics independently, then grouping the evolved scheduling heuristics by some greedy selection strategies, while rarely considering the evolution with ensembles. To the best of our knowledge, no study about GP considers evolution with single individuals and ensembles together in the research area of DFJSS. By allowing breeding between single individuals and ensembles, the search space can be further explored, increasing the likelihood of discovering novel and more effective scheduling heuristics. In order to fill this research gap and improve the performance of GP for DFJSS, we target to propose an effective EGP method.

III. Problem Description and Modelling

The notations and decision variables used in this paper are given in Table I. In DFJSS [37], the shop floor has a set of heterogeneous machines $\mathcal{M} = \{M_1, ..., M_m\}$. A number of jobs $\mathcal{J} = \{J_1, J_2, ..., J_n\}$ arrive at the shop floor dynamically [52]–[58], which is a popular dynamic event in real-world applications. Each job J_i has an arrival time r_i, a weight w_i, a due date d_i, and consists of multiple operations $[O_{i,1}, O_{i,2}, ..., O_{i,p_i}]$ that need to be processed in order. Each operation $O_{i,j}$ has a workload $\pi_{i,j}$, and can be processed by

TABLE I: Notations and decision variables.

Notation	Description
n	The number of jobs
m	The number of machines
J_i	The i-th job
r_i	The arrival time of J_i
w_i	The weight of J_i
d_i	The due date of J_i
p_i	The number of operations J_i consists of
$O_{i,j}$	The j-th operation of the i-th job
$\pi_{i,j}$	The workload of $O_{i,j}$
$\mathcal{M}$	The machine set $\mathcal{M} = \{M_1, ..., M_m\}$
$\mathcal{M}_{i,j}$	The optional machine set for processing $O_{i,j}$, $\mathcal{M}_{i,j} \subseteq \mathcal{M}$
M_k	The k-th machine
γ_k	The processing rate of M_k
$t_{i,j,k}$	The processing time of $O_{i,j}$ on M_k
τ_{k_1,k_2}	The transport time between M_{k_1} and M_{k_2}
$S_{i,j}$	The processing start time of $O_{i,j}$
$C_{i,j}$	The processing completion time of $O_{i,j}$
S_i	The processing start time of J_i
C_i	The processing completion time of J_i
T_i	The tardiness of J_i
L	A sufficiently large constant
z_{i,j,k_1,k_2}	The auxiliary variable domain

an optional machine in $\mathcal{M}_{i,j} \subseteq \mathcal{M}$. Each machine M_k has a unique processing rate γ_k. The processing time $t_{i,j,k}$ of operation $O_{i,j}$ on machine M_k is defined as $t_{i,j,k} = \pi_{i,j}/\gamma_k$. The machines are distributed, and there is a transport time τ_{k_1,k_2} [59] to transport a job between machine M_{k_1} and M_{k_2}.

The goal of DFJSS is to find a schedule that allocates each operation of every job to an available machine while specifying the start time for each operation on its designated machine [60]. This scheduling process must adhere to certain constraints and assumptions, including operation sequencing, non-preemptive behaviour, and so on [61].

In this paper, the objective of DFJSS is to minimise the max-flowtime ($Fmax$), max-tardiness ($Tmax$), and mean-weighted-tardiness ($WTmean$) of all jobs, respectively. The definitions of the three objectives are as follows.

$$Fmax = \max_{i=1}^{n}\{C_i - r_i\}, \quad Tmax = \max_{i=1}^{n}\{T_i\}$$
$$WTmean = \frac{1}{n}\sum_{i=1}^{n}(w_i T_i)$$

where C_i is the completion time of the job J_i in the schedule, and $T_i = \max\{C_i - d_i, 0\}$ is the tardiness of the job J_i.

The problem can be represented as follows [62]–[64]:

Minimise:

$$obj(\cdot),\ obj(\cdot) \in \{Fmax, Tmax, WTmean\} \tag{1}$$

Subject to the following constraints:

$$S_{i,1} \geq r_i \tag{2}$$

$$C_{i,j} \geq S_{i,j} + t_{i,j,k} x_{i,j,k} \tag{3}$$

$$S_{i,j+1} \geq C_{i,j} + \tau_{k_1,k_2} z_{i,j,k_1,k_2} \tag{4}$$

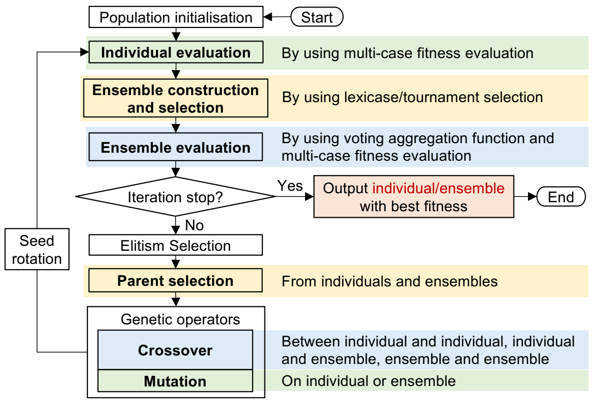
$$S_{i,j} \leq S_{a,b} - t_{i,j,k} + L \cdot (1 - y_{i,j,a,b,k}) \tag{5}$$

$$\textstyle\sum_{k=1}^{m_{i,j}} x_{i,j,k} = 1 \tag{6}$$

$$z_{i,j,k_1,k_2} \leq x_{i,j,k_1} \tag{7}$$

$$z_{i,j,k_1,k_2} \leq x_{i,j+1,k_2} \tag{8}$$

$$z_{i,j,k_1,k_2} \geq x_{i,j,k_1} + x_{i,j+1,k_2} - 1 \tag{9}$$



Start
Population initialisation
Individual evaluation
By using multi-case fitness evaluation
Ensemble construction and selection
By using lexicase/tournament selection
Ensemble evaluation
By using voting aggregation function and multi-case fitness evaluation
Iteration stop?
Yes
Output individual/ensemble with best fitness
End
No
Seed rotation
Elitism Selection
Parent selection
From individuals and ensembles
Genetic operators
Crossover
Between individual and individual, individual and ensemble, ensemble and ensemble
Mutation
On individual or ensemble

Fig. 2: The flowchart of the proposed EGP^e method.

$$z_{i,j,k_1,k_2} \geq 0 \tag{10}$$

$$x_{i,j,k} = \begin{cases} 1, \text{if } M_k \text{ is chosen for } O_{i,j}, \\ 0, \text{otherwise}, \end{cases} \tag{11}$$

$$y_{i,j,a,b,k} = \begin{cases} 1, \text{if } O_{a,b} \text{ is processed after } O_{i,j} \text{ on } M_k, \\ 0, \text{otherwise}, \end{cases} \tag{12}$$

$$z_{i,j,k_1,k_2} = \begin{cases} 1, \text{if } M_{k_1} \text{ is chosen for } O_{i,j} \\ \quad \text{and } M_{k_2} \text{ is chosen for } O_{i,j+1}, \\ 0, \text{otherwise}, \end{cases} \tag{13}$$

where $\forall i, a = 1, \ldots, n$, $\forall j, b = 1, \ldots, p_i$, $\forall k, k_1, k_2 = 1, \ldots, m$. Further, L in Eq. (5) denotes a sufficiently large constant. Constraint (2) ensures that the first operation of each job is not allowed to be processed until the job arrived. Constraint (3) establishes the relationship between the start time $S_{i,j}$ and the completion time $C_{i,j}$ of $O_{i,j}$. Constraint (4) specifies that operation $O_{i,j+1}$ is not allowed to be processed until its preceding operation $O_{i,j}$ has completed and been transported to the selected machine. Constraint (5) states that each machine can only process one operation at a time, and constraint (6) ensures that each operation can only be processed by one of its optional machines. Constraints (7) to (10) define the auxiliary variable domain z_{i,j,k_1,k_2}.

IV. Proposed Method

The flowchart of the proposed EGP^e method is shown in Fig. 2. Population initialisation, fitness evaluation, parent selection, and breeding (crossover and mutation) are main processes in GP, whereas special designs are developed for each part. In addition, the ensemble construction and selection strategy as well as the ensemble evaluation are newly proposed. The detailed population initialisation and the potential combinations of parents for the proposed crossover and mutation operators can be seen in Fig. 3. It illustrates that the population comprises both single individuals and ensembles, and it presents three possible combinations of parents for crossover and two possibilities of parents for mutation. Overall, there are six differences between the proposed EGP^e and classical GP methods:

1) *Population initialisation*: EGP^e initialises and maintains a population containing both single individuals and ensembles instead of only individuals (Section IV-A);

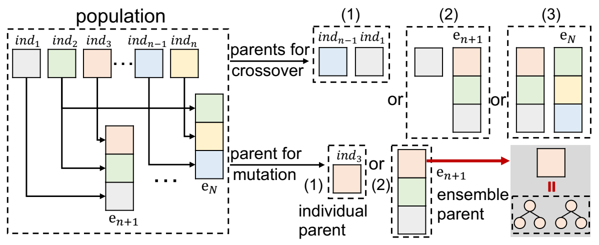


Fig. 3: The population composition and possible parent combination(s) for crossover and mutation of the proposed EGP^e method.

2) *Individual evaluation*: EGP^e uses a multi-case fitness evaluation to assign each individual a list of case-fitnesses and standard fitness rather than only one standard fitness (Section IV-B);
3) *Ensemble Construction and Selection*: EGP^e uses lexicase selection to select individuals to form an ensemble, which is expected to select diverse individuals that are good at handling different cases. Additionally, a newly proposed similarity-checking strategy is used to further ensure the diversity and complementarity of elements in the ensemble (Section IV-C);
4) *Ensemble evaluation*: EGP^e uses a multi-case fitness evaluation to assign each ensemble a list of case-fitnesses and standard fitness (Section IV-D);
5) *Genetic operators*: EGP^e proposes new crossover and mutation operators to generate offspring which allows both individual(s) and ensemble(s) to be parent(s). Also, the generated offspring can be either individual(s) or ensemble(s) (Section IV-E);
6) *Output*: EGP^e allows an individual or ensemble as the final output, depending on which one has the best training performance, while classical GP methods output an individual as the final result and traditional EGP methods output an ensemble as the final result. In other words, the output of EGP^e is more flexible.

A. Population Initialisation

Different from classical GP methods which hold a population of individuals only, our method initialises and maintains a population $\mathbf{P}$ of both single individuals $\Phi = \{ind_1, \cdots, ind_n\}$ and ensembles $\Delta = \{\mathbf{e}_{n+1}, \cdots, \mathbf{e}_N\}$, $\mathbf{P} = \Phi \cup \Delta$, where N represents the population size. During the population initialisation process, n single individuals Φ are first initialised by the ramped-half-and-half method [65]. Following that, the individual evaluation process gives each individual a list of case-fitnesses and standard fitness based on the current training instance by multi-case fitness evaluation [37] (details will be shown in Section IV-B). Then, based on the case-fitnesses of each individual, the ensemble construction and selection process selects diverse individuals to form $N-n$ ensembles Δ (details will be shown in Section IV-C). In this way, we finish the population initialisation using not only individuals but also ensembles to fill the whole population.

B. Individual Evaluation

The process of individual evaluation involves using multi-case fitness evaluation to assign a list of case-fitnesses and standard fitness to each individual in the population [37]. In each generation, a new training instance consisting of m jobs is used for evaluation. To obtain the list of case-fitnesses, the instance is divided into a number of cases based on the jobs. Specifically, if we intend to divide the instance into c cases, the m jobs are split into c groups, with each group containing $g = m/c$ jobs based on their arrival time. This strategy ensures that each case contains the same group of jobs, making it fair to compare the case-fitnesses between individuals. The calculation equation is different according to the objective being optimised.

For example, when considering the mean-flowtime as the objective, the case-fitness $\mathbf{cf}_i(x)$ for the ith case of individual ind_x is computed using Eq. (14). The standard fitness is determined as the mean of the case-fitness values, as depicted in Eq. (15).

$$\mathbf{cf}_i(x) = \frac{1}{g} \sum_{j=g\times(i-1)+1}^{g\times i} (C_j - r_j). \tag{14}$$

$$sf_{mean} = \frac{1}{c} \sum_{i=1}^{c} \mathbf{cf}_i. \tag{15}$$

When targeting the max-tardiness as the objective, the case-fitness $\mathbf{cf}_i(x)$ for the ith case of individual ind_x is calculated using Eq. (16). The standard fitness is computed as the maximum value among the case-fitnesses, as presented in Eq. (17).

$$\mathbf{cf}_i(x) = \max_{j\in\{g\times(i-1)+1,\ldots,g\times i\}} T_j. \tag{16}$$

$$sf_{max} = \max\{\mathbf{cf}_1, \mathbf{cf}_2, \cdots, \mathbf{cf}_c\}, \tag{17}$$

The case-fitnesses are used for lexicase selection, while the standard fitness is used for elitism selection and tournament selection. More information about the multi-case fitness evaluation can be found in [37].

Algorithm 1: Ensemble construction and selection.

```
Input: Individual set: $\Phi$; Ensemble size: $s$; Number of cases for
       similarity-checking: $h$; Similarity threshold: $\zeta$.
Output: An ensemble of scheduling heuristics: $\mathbf{e}_i$.
$\mathbf{e}_i \leftarrow \emptyset$;
$times \leftarrow 0$;
while $size(\mathbf{e}_i) < s$ & $times < 3 * s$ do
    $ind_a \leftarrow LexicaseSelection(\Phi)$;
    $o_a \leftarrow CaseOrder(ind_a)$;
    // Similarity-checking
    $\eta_{a,b} \leftarrow 0$;
    for $ind_b \in \mathbf{e}_i$ do
        $o_b \leftarrow CaseOrder(ind_b)$;
        for $k = 1 \to h$ do
            if $o_a[k] == o_b[k]$ then
                $\eta_{a,b} \leftarrow \eta_{a,b} + \frac{c-k-1}{c} \cdot P(o_a[k], o_b[k])$;
            end
        end
    end
    if $\eta \leq \zeta$ then
        $\mathbf{e}_i \leftarrow \mathbf{e}_i \cup ind_a$;
    end
    $times \leftarrow times + 1$;
end
return $\mathbf{e}_i$;
```

C. Ensemble Construction and Selection

The *ensemble construction and selection* process selects individuals to construct ensembles. The goal of ensemble construction and selection is to choose diverse and complementary individuals for constructing each ensemble. It involves two parts: lexicase selection and similarity-checking. The pseudo-code of forming an ensemble by the ensemble construction and selection is shown in Algorithm 1. At the beginning, the ensemble $\mathbf{e}_i$ is empty (line 1). To achieve diverse and complementary individuals for an ensemble, we first use lexicase selection to select individuals [37] (line 4). By lexicase selection, we expect to select a diverse set of expert individuals that are good at handling different cases. Every time a candidate individual is selected and ready to be added to the ensemble, the similarity-checking process is triggered to check whether the selected candidate individual is good at handling different cases from the individuals that are already in the ensemble.

To be specific, as the individual is selected by lexicase selection through a sequence of cases, it is expected to perform well on the cases that rank front in the case sequence. For example, we consider 5 cases and select an individual ind_a based on the case sequence $o_a = [2, 1, 5, 3, 4]$. Among these 5 cases, this individual is expected to perform the best on the 2nd case (as $o_a[1] = 2$), followed by the 1st case (as $o_a[2] = 1$), and finally the 4th case (as $o_a[5] = 4$). Further, we believe that different cases have different importances. In this case, every time an individual is added to the ensemble, the case sequence o_a used for the lexicase selection is also recorded. Then, we check the similarity between the top h cases of the case sequence of the candidate individual and that of all the individuals in the ensemble (line 11). Only when the individual is selected using different cases (i.e., the similarity η is smaller than a threshold ζ), it can be added to the ensemble (line 16). Here, the threshold ζ is set to 0, which means that we expect elements in the ensemble to be selected based on totally different top h cases of the case sequence.

For example, if we consider that the first 2 cases ($h = 2$) play more important roles, then we perform the similarity-checking between individuals by comparing whether the first 2 cases are the same. Meanwhile, if we set the ensemble size to s, then when there have been s individuals added to the ensemble or $3 * s$ individual additions have been tried, the process is finished and an ensemble is obtained. Specifically, the similarity $\eta_{a,b}$ between individuals ind_a and ind_b is calculated as Eq. (18).

$$\eta_{a,b} = \sum_{k=1}^{h} \frac{c-k-1}{c} \cdot P(o_a[k], o_b[k]) \tag{18}$$

where $P(o_a[k], o_b[k])$ is a decision variable, which is 1 if the kth case $o_a[k]$ of individual ind_a equal to the kth case $o_b[k]$ of individual ind_b, and 0 otherwise.

D. Ensemble Evaluation

At each generation, both single individuals and ensembles are evaluated on the same training instance(s). Since an ensemble consists of multiple individuals (i.e., scheduling heuristics), an aggregation method is required to make a decision at each decision point. In this paper, we adopt the voting strategy [24] to make the final decision. The voting strategy selects the operation/machine that received the most votes from all the scheduling heuristics in the ensemble. Same to the individual evaluation, we use the multi-case fitness evaluation [37] to evaluate each ensemble. After ensemble evaluation, the standard fitness and a list of case-fitnesses of each ensemble are obtained.

E. Genetic Operators

During the evolutionary process, tournament selection is used to select parents, which allows both single individuals and ensembles to be chosen based on their standard fitnesses. Crossover and mutation operators are commonly used to generate offspring in GP. We develop novel crossover and mutation operators, which can be applied to both single individuals and ensembles to generate offspring. In addition, in order to explore a wider range of ensemble combinations, the proposed ensemble construction and selection in Section IV-C enables the generation of ensemble offspring by selecting individuals from the current generation to form ensembles. The details about the newly proposed crossover and mutation operators are shown as follows.

1) Crossover: Crossover requires two parents to generate offspring. In our proposed EGP^{e} method, the selected two parents have three possible combinations, i.e., individual and individual, individual and ensemble, or ensemble and ensemble:

- If the two parents are both individuals, the conventional tree swapping crossover [66] is adopted to generate two offspring, and the two offspring are individuals.
- If one parent is an individual and the other is an ensemble, then one offspring is generated by randomly selecting an individual from the ensemble to do tree swapping crossover between this individual and the other individual parent. The generated offspring is an individual. The other offspring is generated by randomly replacing an individual of the ensemble with the individual parent. This offspring is an ensemble. The detailed process is shown in Fig. 4.
- If the two parents are both ensembles, then two offspring are generated by randomly selecting an individual from each ensemble and swapping them. Both offspring are ensembles. The detailed process is shown in Fig. 5.

2) Mutation: Mutation generates offspring from a single parent. There are two situations depending on the parent, individual or ensemble:

- If the parent is an individual, then the standard subtree mutation [66] is adopted to generate offspring, and the offspring is also an individual.
- If the parent is an ensemble, then we randomly select an individual from the ensemble and apply the standard subtree mutation to generate an offspring. The generated offspring is an individual. The detailed process is illustrated in Fig. 6.

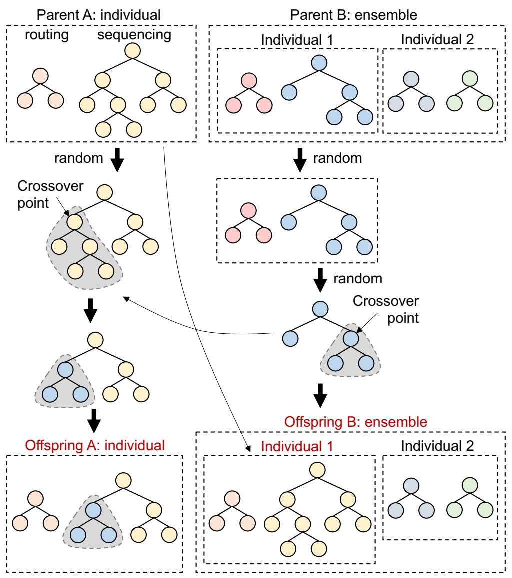


Fig. 4: The crossover process between an individual and an ensemble.

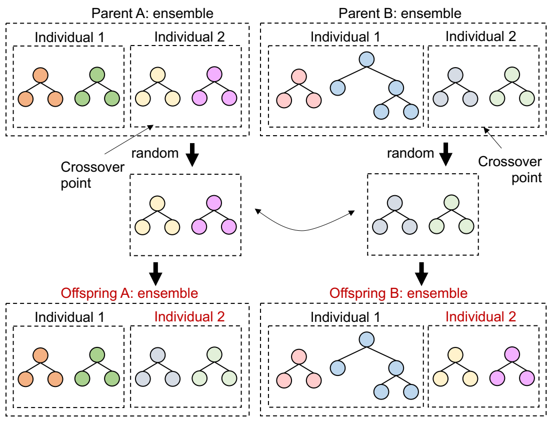


Fig. 5: The crossover process between two ensembles.

Note that for mutation, the generated offspring is a single individual regardless of whether the parent is an individual or an ensemble. The reason for allowing only individuals rather than ensembles as offspring is that in addition to producing ensemble offspring that can be inherited to the next generation, we also want to explore more ensemble structures by randomly combining the generated offspring to form different ensembles based on the proposed ensemble construction and selection strategy. Therefore, in order to leave some place for new ensembles while maintaining the ratio of single individuals and ensembles in the population at the same time, only individuals are allowed as offspring for mutation. In addition, to further maintain the same ratio of single individuals and ensembles in the population, the proposed method limits the number of ensembles to a fixed number $N - n$. That is, when the generated ensembles reach $N - n$, the proposed method ignores the generated ensembles and only keeps the generated individuals as offspring.

V. Experiment Design

The DFJSS simulation [41] is used to simulate different job shop scheduling systems with varying difficulties. We consider a DFJSS problem with 10 heterogeneous machines, whose processing rates are sampled from the range $[10, 15]$. The

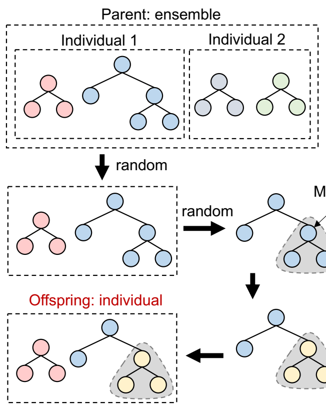


Fig. 6: The mutation process for an ensemble parent.

machines are located at different positions on the shop floor. When transferring operations from one machine to another, the transportation time between machines is sampled from the range $[7, 100]$. We consider dynamic and large-scale DFJSS scenarios with 6000 jobs continuing to arrive at the shop floor. The arrival time of the jobs follows a Poisson process, i.e., the gap between subsequent arrivals follows an exponential distribution. Also, for each job, a due date is assigned to 1.5 times of its processing time plus arrival time. The importance of each job is represented by its weight. The higher the weight, the more important a job is. In the experiments, 20%, 60%, and 20% of the jobs are given weights 1, 2, and 3, respectively. To be noticed, the first 1000 jobs are used as warm-up jobs to obtain a stable scheduling system. Each job consists of a number of operations to be completed in sequence. For each job, the number of operations is sampled between 2 and 10. Each operation has a workload, which is sampled from the range $[100, 1000]$. The processing time of an operation by a machine is then calculated by dividing the workload of the operation by the processing rate of the machine.

In the simulation, the utilisation level is an important factor to reflect how busy the job shop is. A higher utilisation level denotes a busier job shop. In this work, we consider six scenarios by considering two different utilisation levels (0.85 and 0.95) and three different objectives ($Fmax$, $Tmax$, and $WTmean$). For example, the scenario <Fmax,0.85> represents a DFJSS simulation considering the objective max-flowtime ($Fmax$) and a utilisation level of 0.85.

To validate the effectiveness of the presented EGP^e method, this paper compares it against the following traditional and ensemble GP algorithms: GP [66], Bagging GP (BagGP) [49], cooperative coevolution GP (CCGP) [34], DivNichGP [35], multidimensional multiclass GP with multidimensional populations (M3GP) [67], and ensemble GP (eGP) [36]. The details about these comparison methods are shown in the supplementary file. The reasons to select these algorithms as the comparison methods are because: 1) GP is a typical method to the DFJSS problem, and is widely used as a benchmark in this field; 2) BagGP is a variation that applies a classical ensemble learning technique (Bagging) to GP; 3) CCGP, DivNichGP, M3GP, and eGP are effective GP-based methods for evolving ensembles of heuristics.

Table II exhibits the terminals and functions utilised for

TABLE II: The description of terminals and functions.

Terminal	Description
TIS	time stay in the system = t - arrivalTime
W	the job weight
NOR	the remaining operation number of the job
WKR	the remaining work
rDD	the relative due date = DD - t
SL	the slack
PT	the processing time for the operation
OWT	the waiting time for the operation = t - ORT
NPT	the median processing time for the succeeding operation
MWT	the waiting time for the machine = t - MRT
NIQ	the operation number in the machine waiting queue
WIQ	the total work in the machine waiting queue
TRANT	the transportation time
Functions	$max, min, +, -, \times, protected\ /$

* t: current time; arrivalTime: job arrival time; DD: due date; ORT: ready time of operation; MRT: ready time of machine;

TABLE III: The configuration of parameters for GP methods.

Parameter	Value
Initialisation method	Ramped half and half
Initial maximum/minimum tree depth	6 / 2
Population size	1000
Maximal number of evaluations	50000
Maximal tree depth	8
Elitism individual/ensemble	8 / 2
Crossover/mutation rate	0.8 / 0.2
Non-terminal/terminal selection rate	0.9 / 0.10
Parent selection	Tournament selection
Ensemble construction and selection	Tournament selection /Lexicase selection
Ensemble from offspring probability	0.3

constructing individuals in this work. The terminals encompass attributes linked to jobs (i.e., TIS, W, NOR, and WKR), operations (i.e., rDD, SL, PT, OWT, and NPT), machines (i.e., MWT, NIQ, and WIQ), and transportation (i.e., TRANT). Regarding the functions, the operators max and min each require two arguments and produce the maximum and minimum values among these arguments, respectively. The arithmetic operators accept dual arguments. The "/" operator is protected, yielding a result of 1 when divided by zero. The parameter configurations for all GP methods can be found in Table III. Specifically, an ensemble size of 5 is set for EGP^e, as previous research has demonstrated that this ensemble size is able to provide promising performance, and increasing the ensemble size does not yield significant improvements in results [21], [68].

VI. Experimental Results

A. Sensitivity Analysis

To examine the impact of the number of ensembles on the proposed EGP^e method, a sensitivity analysis was conducted. Fig. 7 presents violin plots illustrating the test performance of 30 runs of the proposed EGP^e method with varying numbers of ensembles (20, 40, 60, 80, and 100) in the population across six scenarios. To be noticed, the number of single individuals in the population changes as the number of ensembles changes. For example, when we consider 20 ensembles, the number of single individuals in the population will be $1000 - 20 \times 5 = 900$. The black curves in the figure represent the trend of the mean test performance as the number of ensembles increases. It is evident that the effect of the number of ensembles on the test performance varies across different scenarios.

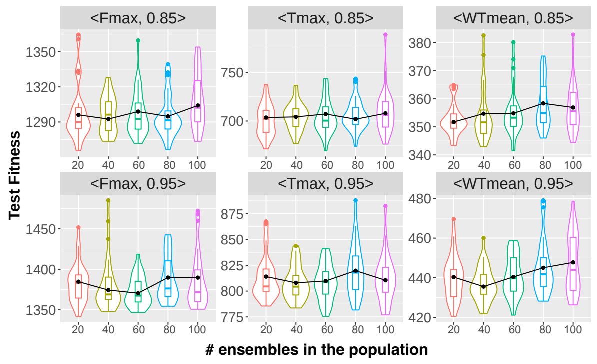


Fig. 7: The violin plots of the test performance of 30 runs of the proposed EGP^e with different ratios of single individuals and ensembles on 6 scenarios.

Moreover, Table IV provides the mean and standard deviation of the test performance for EGP^e using different numbers of ensembles in the population across 30 independent runs in the six scenarios. To rank the EGP^e method with varying numbers of ensembles, we conducted the Friedman test [69]. The results indicate that the EGP^e method with 40 ensembles achieved the highest rank, followed by 60, 20, 80, and 100 ensembles, respectively. Hence, we observe that the EGP^e method with 40 ensembles delivers the best performance among the tested parameters. Subsequent sections will focus on further analyses using the EGP^e method with 40 ensembles. To simplify the description, in the subsequent sections we use EGP^e to denote EGP^e with 40 ensembles.

B. Test Performance

Table V gives the mean and standard deviation of the test performance derived from 30 independent runs of the EGP^e method and compared methods across six DFJSS scenarios. We employ the Wilcoxon rank sum test [70] to perform comparisons, where EGP^e is compared with GP, BagGP, CCGP, DivNichGP, M3GP, and eGP, respectively. The results are considered significant when the obtained p-value is less than 0.05 and a smaller (better) mean test performance is given. The symbols "↑/↓/=" next to the results of EGP^e in the table indicate whether the results are significantly better than, worse than, or comparable to the results of each algorithm located to the left of EGP^e. In the case of the <Fmax, 0.85> scenario (first row of the table), the (=)(↑)(↑)(↑)(↑)(↑) for EGP^e indicates that EGP^e shows no significantly different performance from that of GP, performs significantly better than BagGP, CCGP, DivNichGP, M3GP, and eGP. In addition, the Friedman test [69] is conducted to rank these methods.

From Table V, it is observed that EGP^e performs significantly better than DivNichGP on two scenarios and is comparable to DivNichGP on the remaining four scenarios. Compared to standard GP, EGP^e outperforms it on three out of six scenarios. For the remaining three scenarios, EGP^e performs similarly to GP but with better mean test performance and smaller standard deviation. Moreover, EGP^e significantly outperforms BagGP, CCGP, M3GP, and eGP on all six scenarios. In addition, the outcomes of the Friedman test presented in Table V indicate that EGP^e ranked first, followed by compared methods with the order DivNichGP, GP, eGP, BagGP, CCGP,

TABLE IV: The mean and standard deviation test performance of the EGP^e method with different numbers of ensembles in the population through 30 independent runs across 6 scenarios.

Scenario	EGP^e-20	EGP^e-40	EGP^e-60	EGP^e-80	EGP^e-100
$< Fmax, 0.85 >$	1297.57(24.33)	1296.30(15.68)	1296.95(18.93)	1294.92(17.83)	1307.81(23.21)
$< Fmax, 0.95 >$	1382.50(24.17)	1380.17(32.67)	1372.36(18.27)	1387.91(27.49)	1385.42(34.20)
$< Tmax, 0.85 >$	700.98(16.23)	704.40(13.83)	704.29(17.65)	705.48(15.23)	709.33(21.96)
$< Tmax, 0.95 >$	812.95(20.88)	807.93(16.10)	808.15(17.46)	819.59(24.45)	812.00(22.79)
$< WTmean, 0.85 >$	352.60(5.69)	354.06(9.25)	355.33(8.71)	358.09(8.45)	357.10(8.93)
$< WTmean, 0.95 >$	440.00(11.06)	436.66(8.59)	441.02(10.15)	445.02(13.91)	447.20(14.51)
Average rank	2.5	**1.83**	2.33	4	4.33

TABLE V: The mean and standard deviation performance of the EGP^e and compared methods on unseen test instances through 30 independent runs across 6 scenarios.

Scenario	GP	BagGP	CCGP	DivNichGP	M3GP	eGP	EGP^e
$< Fmax, 0.85 >$	1304.36(23.96)	1338.32(20.62)	1354.59(36.46)	1305.82(15.66)	1558.49(72.74)	1322.48(25.00)	1296.30(15.68)(=)(↑)(↑)(↑)(↑)(↑)
$< Fmax, 0.95 >$	1388.71(27.11)	1446.07(31.94)	1472.51(87.37)	1389.39(27.44)	1768.12(298.25)	1418.06(52.46)	1380.17(32.67)(↑)(↑)(↑)(↑)(↑)(↑)
$< Tmax, 0.85 >$	711.58(14.85)	742.77(13.18)	768.02(40.37)	705.16(14.90)	918.16(54.20)	722.19(24.10)	704.40(13.83)(↑)(↑)(↑)(=)(↑)(↑)
$< Tmax, 0.95 >$	816.22(18.08)	874.25(12.31)	899.91(71.35)	813.72(20.42)	1082.08(99.83)	833.77(23.87)	807.93(16.10)(↑)(↑)(↑)(=)(↑)(↑)
$< WTmean, 0.85 >$	354.89(9.53)	370.13(6.23)	361.58(14.27)	352.62(7.33)	430.99(13.87)	358.15(8.29)	354.06(9.25)(=)(↑)(↑)(=)(↑)(↑)
$< WTmean, 0.95 >$	440.69(10.79)	457.32(5.26)	446.36(10.04)	439.35(9.10)	531.80(25.51)	443.74(11.48)	436.66(8.59)(=)(↑)(↑)(=)(↑)(↑)
Improvement	0.73%	4.88%	5.56%	0.40%	20.52%	2.16%	-
Average rank	2.67	5.33	5.67	2.17	7.0	4.0	**1.17**

and M3GP. More precisely, the average improved percentage ρ of the proposed EGP^e over each comparison method A across the 6 scenarios are calculated based on the Eq. (19) and shown at the bottom of Table V. The reason why BagGP, CCGP, M3GP, and eGP perform worse than the proposed EGP^e and GP is that these methods do not consider the problem-specific characteristics of DFJSS. They neglect the commonly used seed rotation strategy in DFJSS, which plays a crucial role in its optimisation process.

$$\rho(EGP^e|A) = \frac{1}{6}\sum_{i=1}^{6} \frac{(Obj(A) - Obj(EGP^e))}{Obj(A)} \tag{19}$$

To further compare the proposed EGP^e with standard GP and DivNichGP which are the second and third places, Fig. 8 shows their convergence curves of the mean test performance of 30 runs from the number of evaluations 5000 to the number of evaluations 50000 on 6 scenarios. As we can see, compared to standard GP, EGP^e converges faster and ultimately converge to better results on scenarios <Tmax,0.85>, <Fmax,0.95>, and <Tmax,0.95>. For the remaining scenarios, the curves of standard GP and EGP^e are quite close to each other but EGP^e finally gives smaller (better) test performance when arriving at 50000 evaluations. Compared to DivNichGP, except for the scenario <WTmean,0.85>, EGP^e converges faster and gives better test performance on most of the generations on the remaining scenarios. Overall, the results confirm that the proposed EGP^e method is more effective than standard GP, BagGP, CCGP, DivNichGP, M3GP, and eGP.

C. Effectiveness of Ensemble Construction and Selection

To study the effect of the proposed ensemble construction and selection on the proposed method, we compare the proposed EGP^e method to a variation of EGP^e that uses the standard **t**ournament selection to select elements to form ensembles (not using the proposed ensemble construction and selection), which is named EGP^t. Table VI provides the mean and standard deviation test performance resulting from 30 independent runs of both EGP^t and EGP^e across 6

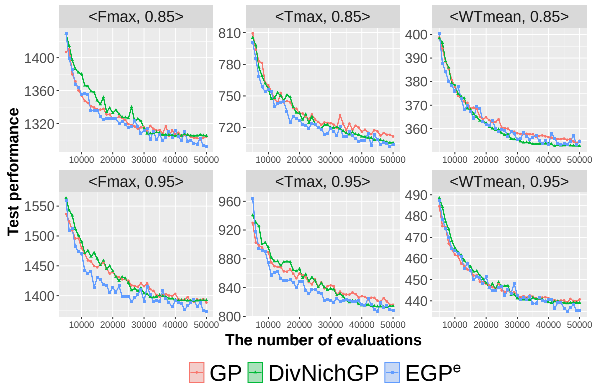


Fig. 8: The convergence curves of the mean test performance of 30 runs of standard GP, DivNichGP, and EGP^e on 6 scenarios.

TABLE VI: The mean and standard deviation of test performance from 30 independent runs of **EGP^t** and **EGP^e** across 6 scenarios.

Scenario	EGP^t	EGP^e
$< Fmax, 0.85 >$	1306.09(23.11)	1296.30(15.68)(=)
$< Fmax, 0.95 >$	1421.66(184.43)	**1380.17(32.67)**(↑)
$< Tmax, 0.85 >$	710.56(16.87)	704.40(13.83)(=)
$< Tmax, 0.95 >$	820.41(21.94)	**807.93(16.10)**(↑)
$< WTmean, 0.85 >$	354.21(9.22)	354.06(9.25)(=)
$< WTmean, 0.95 >$	440.41(8.94)	**436.66(8.59)**(↑)

scenarios. Notably, EGP^e outperforms EGP^t on 3 scenarios and demonstrates no statistical difference in comparison to EGP^t within the remaining 3 scenarios. Moreover, on the 3 scenarios with similar performance, the proposed EGP^e method gives numerically better performance (smaller mean test objective value) than EGP^t. Through the comparison of these two methods, the effectiveness of the proposed ensemble construction and selection strategy is verified.

D. Effectiveness of Genetic Operators

To test the effectiveness of the presented genetic operators that consider single individuals and ensembles together, we conducted experiments comparing EGP^e to a variation of EGP^e with **c**lassical crossover and mutation operators, which

TABLE VII: The mean and standard deviation of test performance from 30 independent runs of **EGP**^{c} and **EGP**^{e} across 6 scenarios.

Scenario	EGP^c	EGP^e
$< Fmax, 0.85 >$	1302.21(25.29)	1296.30(15.68)(=)
$< Fmax, 0.95 >$	1388.67(31.58)	1380.17(32.67)(=)
$< Tmax, 0.85 >$	707.28(13.74)	704.40(13.83)(=)
$< Tmax, 0.95 >$	820.45(24.65)	**807.93(16.10)**(↑)
$< WTmean, 0.85 >$	357.16(7.86)	**354.06(9.25)**(↑)
$< WTmean, 0.95 >$	440.69(12.11)	436.66(8.59)(=)

we named EGP^c. The EGP^c method only allows individuals to be parents and produce individual offspring, while the proposed ensemble construction and selection strategy is still used to populate the ensemble portion of the population. Table VII presents the mean and standard deviation test performance by 30 runs of EGP^c and EGP^e on 6 scenarios. The results show that EGP^e outperforms EGP^c on 2 scenarios and performs comparably on the other 4 scenarios. These findings confirm the effectiveness of the presented genetic operators (new crossover and mutation) in generating high-quality offspring by leveraging the advantages of both single individuals and ensembles.

VII. Further Analyses

A. Performance of Ensemble and Elements

We typically expect an ensemble to perform better than each individual element within it. In this case, to analyse the performance relationship between the ensemble and its individual elements, we create the scatter plots (x, y) with the ensemble performance as the x and the element performance as the y. The visualisation is based on the 30 runs from the number of evaluations 40000 to 50000, where EGP^e produces the ensemble as the best solution for every 1000 evaluations. The scatter plots of the training performance of the ensemble versus the training performance of the element from the number of evaluations 40000 to 50000 of 30 runs by EGP^e on 6 scenarios are shown in Fig. 9. To be noticed, the points in red colour represent that the ensemble outperforms the corresponding individual element and the line denotes the reference line of $y = x$ which makes it easy to see the performance relationship between the ensemble and its individual elements. It can be seen from Fig. 9, the learned ensembles always outperform their individual element on the training instances.

However, we observe different phenomena on unseen instances. Fig. 10 shows the scatter plots of the test performance of the ensemble versus the test performance of the element from the number of evaluations 40000 to 50000 of 30 runs by EGP^e on 6 scenarios. Similarly, the points in red colour represent that the ensemble outperforms the corresponding individual element, while the points in blue colour indicate that the ensemble performs worse than the corresponding individual element. As we can see, it is possible for an individual element to outperform the ensemble it belongs to on the unseen test set. Nevertheless, the visualisation indicates that in most cases, the ensemble still outperforms the individual element by a significant margin.

This finding emphasises the importance of carefully selecting elements to ensure that the formed ensemble can achieve better performance. This also suggests that a combination of single individuals and ensembles is necessary to achieve good performance. Moreover, it suggests that the ensemble construction and selection process may not fully capture the underlying characteristics of the problem for the unseen data, which could be explored in future research to improve the ensemble construction and selection strategy.

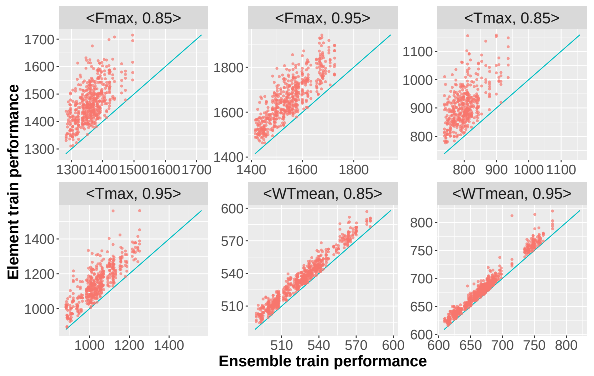


Fig. 9: The scatter plots of the train performance of the ensemble versus that of its each element from the number of evaluations 40000 to 50000 of 30 runs by **EGP**^{e} on 6 scenarios.

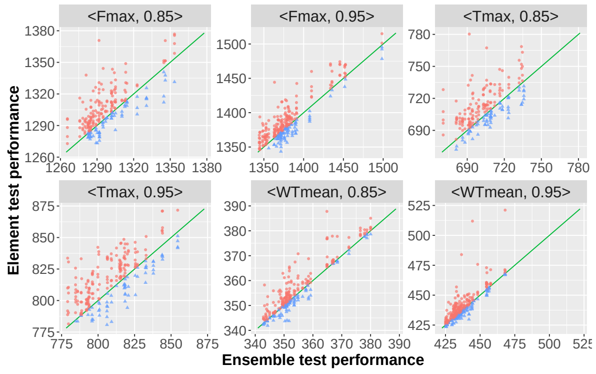


Fig. 10: The scatter plots of the test performance of the ensemble versus the test performance of its each element from the number of evaluations 40000 to 50000 of 30 runs by **EGP**^{e} on 6 scenarios.

B. Ensemble Win Percentage

The proposed method incorporates both single individuals and ensembles in its population, it is interesting to track which type of solution, i.e., individual or ensemble, performs better at each generation, as well as the percentage of times each type of solution achieves the best performance across 30 runs. Fig. 11 illustrates the percentage of times an ensemble achieves the best performance every 1000 evaluations of EGP^c and EGP^e across six scenarios of 30 runs. For different scenarios, we observe different phenomena. For the scenarios with max-objective, at the beginning, the percentage of ensembles that achieve the best solution is low, but as evolution progresses, the percentage of ensembles that achieve the best solution increases, reaching a peak of approximately 50% around at sixth generation. Subsequently, the percentage of ensembles achieving the best solution decreases but remains stable at around 10% towards the end of evolution. For the scenarios with mean-objective, a similar trend in the curve was observed, but the corresponding values were relatively large. To be specific, initially, the proportion of ensembles that reach

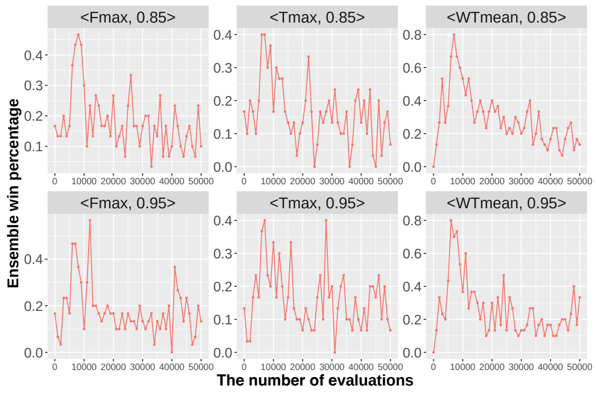


Fig. 11: The convergence curves of the percentage of the ensemble is output as the best solution every 1000 evaluations on 6 scenarios of 30 runs by the proposed EGP^e.

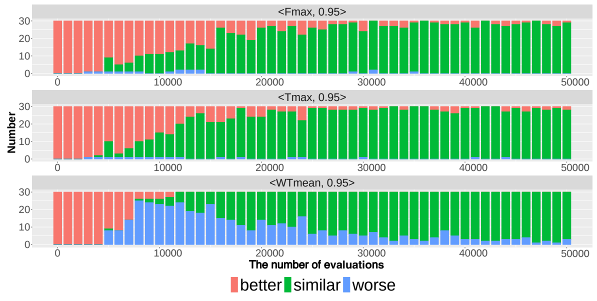


Fig. 12: The number of runs that ensembles are significantly better and worse than, or comparable to individuals out of 30 runs every 1000 evaluations on 3 scenarios of **EGP^e**.

the best solution is low. However, as evolution continues, this proportion gradually increases, peaking at approximately 80% by the sixth generation. Subsequently, the percentage of ensembles attaining the best solution declines but stabilises at around 20% towards the end of the evolutionary process. This is a high proportion considering that the proportion of ensembles in the entire population is only 4.8%. By tracking the performance of each solution type and the percentage of times each type achieves the best performance, we suggest that a dynamic adjustment strategy to change the proportion of ensembles and individuals in the population can help to optimise the population and improve overall solution quality, which can be explored in the future. In addition to this, the above results show that the ensemble has a more pronounced role on the scenarios with mean-objective. In this case, we suggest to further analyse the difference between different objectives and design specific strategies for different objectives.

Furthermore, to provide a more intuitive understanding of the results, we present the number of runs in which ensembles are significantly better, worse, or comparable to individuals in the population out of 30 runs every 1000 evaluations on six scenarios of EGP^e, which is shown as Fig. 12. It can be seen that the distinction is not only scenario-dependent but also changes with the evolutionary process. To be specific, in scenarios with a max-objective, ensembles outperform individuals in most cases during the early stages of evolution. As the evolution progresses, the frequency of ensembles outperforming individuals decreases, while the frequency of ensembles behaving similarly to individuals increases. In the late stages of evolution, ensembles perform significantly better than individuals in a few cases and have no statistical difference from individuals in most cases. However, in the scenario with a mean-objective (i.e., the bottom graph in Fig. 12), ensembles perform better than individuals in most cases at the early stages of evolution. In the middle stages of evolution, ensembles mostly perform significantly worse than individuals, while in the late stages of evolution, ensembles exhibit no significant difference from individuals in most cases.

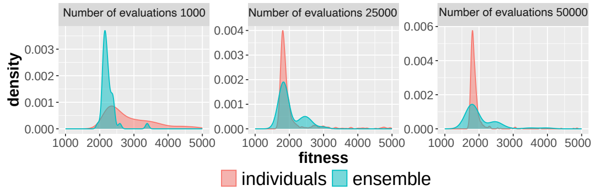


Fig. 13: The fitness distribution of ensembles and individuals of a single run at the beginning (number of evaluations 1000), middle (number of evaluations 25000), and late (number of evaluations 50000) stages of evolution on the scenario **<Fmax,0.95>** of **EGP^e**.

More detailed, Fig. 13 illustrates the fitness distribution of single individuals and ensembles from a single run at the early (the number of evaluations 1000), middle (the number of evaluations 25000), and late (the number of evaluations 50000) stages of evolution of EGP^e on the <Fmax,0.95> scenario. The range of fitness values displayed in the figure is limited to 1000 to 5000, as most single individuals and ensembles fall within this range. As observed in Fig. 13, at the beginning stage of evolution, the shape of the distribution of fitness of ensembles is higher and leaner than individuals and performs better overall, while as the evolution process goes on, the shape of the distribution of fitness of ensembles becomes shorter and wider than that of individuals. At the same time, the overall density of ensembles' fitness is relatively uniformly distributed, rather than being very concentrated within a certain range, as is the case with individuals. Given the above analyses, we obtain the following observations:

1) Firstly, even in the early stages of evolution, when individuals tend to perform poorly, ensembles can still achieve better performance by combining them;
2) Secondly, the better performance achieved by ensembles can help to provide a lower bound for optimisation and thus is able to improve the overall population performance of both single individuals and ensembles;
3) Thirdly, the wider and shorter density distribution of ensemble fitness as evolution proceeds indicates that ensembles can provide more stable performance than individuals;
4) Finally, when focusing on the proportion of individuals/ensembles that perform exceptionally well (i.e., fitness within the range of 1000 to 2000), ensembles still have a higher proportion than individuals. This observation further confirms the effectiveness of ensembles as an effective technique for improving solution quality.

C. Elements Contribution to Ensemble

To avoid the situation that one individual dominates all the others or the situation that other individuals make no contributions to the ensemble, it is interesting to study the

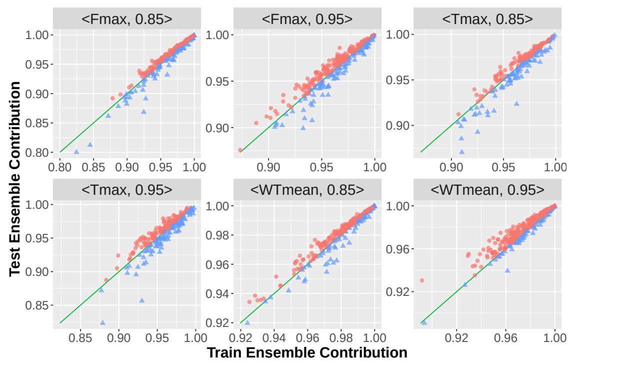


Fig. 14: The scatter plots of training ensemble contribution versus test ensemble contribution of each individual in the learned ensembles from the number of evaluations 40000 to 50000 by the proposed **EGP**^{e} on 6 scenarios of 30 runs.

ensemble contribution of each individual in an ensemble. Here, the *ensemble contribution* δ_i represents the percentage of the same decisions made by an individual element ind_i compared to the decisions made by the ensemble e_j across all decision points [25], which can be calculated by Eq. (20).

$$\delta_i = \frac{\sum_{d=1}^{D_j} Z_{i,j,d}}{D_j} \tag{20}$$

where D_j represents the total number of decisions when e_j works for the training instance(s). $Z_{i,j,d}$ is a decision variable, equal to 1 when ind_i gives the same decision with e_j on the dth decision point.

If an individual has an ensemble contribution of 0 (no same decision), it means that the individual does not make any contribution to the ensemble. Conversely, if an individual has an ensemble contribution of 1, it implies that this individual is dominating the ensemble. We expect each individual to give a high ensemble contribution, also it is expected that each individual gives a relatively consistent (similar) ensemble contribution on training instances (training ensemble contribution) and on unseen test instances (test ensemble contribution). Fig. 14 gives the point plots of training ensemble contribution versus test ensemble contribution of each individual in an ensemble when the ensemble is output as the best solution at each generation by the proposed EGP^e on 6 scenarios of 30 runs. The points in red colour represent that the test ensemble contribution is higher than the training ensemble contribution, while the points in blue colour indicate that the test ensemble contribution is lower than the training ensemble contribution. Also, the line denotes the reference line of $y = x$, which makes it easy to see the relationship between the training ensemble contribution and the test ensemble contribution of each individual. It can be seen that, EGP^e can evolve ensembles wherein each element can support a high training and test ensemble contribution (higher than 0.89). This gives evidence that every element in ensembles contribute to the performance of the ensemble. Also, about half of the situations where the training ensemble contribution is higher than the test ensemble contribution, and half of the situations where it is the opposite. In general, the figure shows that each point lies near the line $y = x$, which means each individual can provide a test ensemble contribution that is generally consistent with its training ensemble contribution. This finding suggests that we can trust the training results and use the trained ensembles on unseen instances.

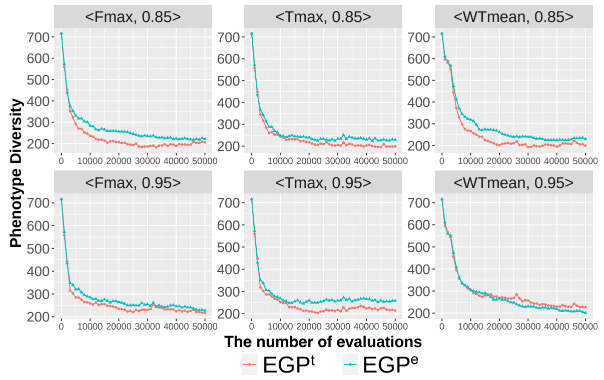


Fig. 15: The convergence curves of the **phenotypic diversity** of the individuals in the population by the proposed **EGP**^{e} and **EGP**^{t} on 6 scenarios of 30 runs.

D. Diversity

Diversity plays an important role not only in the population but also in the ensemble. In this paper, the proposed method holds a population containing both single individuals and ensembles, and most of the ensembles are formed by individuals in the population using the developed ensemble construction and selection strategy. This section explores the diversity among individuals within the population. Within the domain of DFJSS, *phenotypic diversity* is more meaningful than *genotypic diversity*, as scheduling heuristics that have different structures (genotypes) can have the same behaviour (phenotype). Here, the phenotypic diversity is quantified through the calculation of the distinct sets of decisions made by individuals within the population across 20 sequencing and 20 routing decision points [41]. Fig. 15 shows the convergence curves of the phenotypic diversity on each generation by the proposed EGP^e and EGP^t on 6 scenarios of 30 runs. It can be observed that the proposed EGP^e gives a higher phenotypic diversity on almost all the generations on 5 of the scenarios, except for the scenario <WTmean,0.95>. Analysing these results with the test performance of EGP^e and EGP^t from Section VI-C, it shows that higher phenotypic diversity is able to contribute to obtaining good performance on the scenarios with max-objective, while not contribute to obtaining good performance on the scenario with mean-objective.

This observation implies that, for the DFJSS problem, it is beneficial to prioritise increasing diversity when optimising the max-objective, while focusing more on convergence when optimising the mean-objective. This is because of the characteristic of these two types of objectives in DFJSS: 1) The max-objective is more sensitive to outliers, as it focuses on optimising the worst-case job/machine. If there are occasional jobs or machines that have significantly higher processing times, the max-objective may prioritise reducing the impact of those outliers, potentially at the expense of the majority of jobs. Consequently, there is a higher risk of getting trapped in local optimal when handling the max-objective. 2) The mean-objective, on the other hand, is less sensitive to outliers and

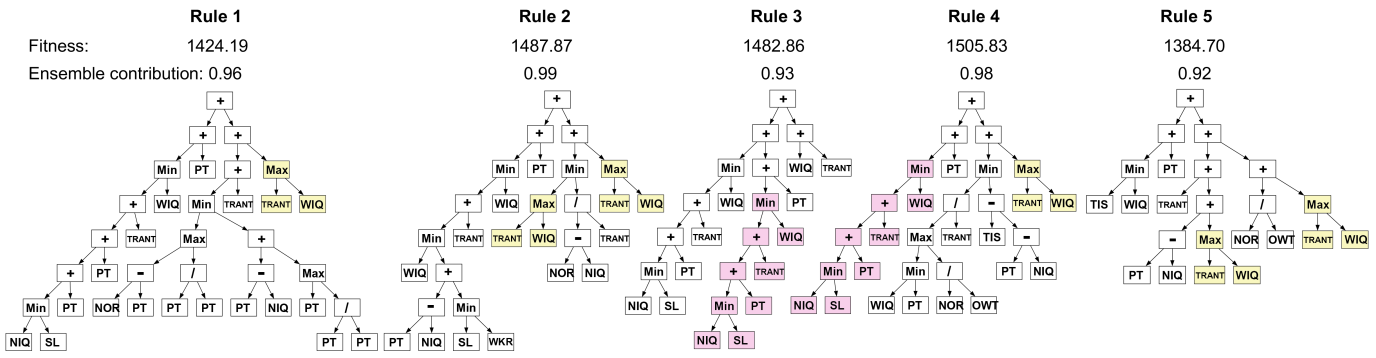


Fig. 16: The tree structure of routing rule of an evolved ensemble with its training fitness and training ensemble contribution of a single run on scenario <Fmax,0.85> of **EGP**^{e}.

extreme values. It aims for a balanced performance across all jobs and machines. By converging towards solutions that distribute the workload evenly, the algorithm can ensure that no particular job or machine is significantly overburdened. This leads to improved overall system performance and minimises potential bottlenecks or delays. In summary, when optimising the max-objective, diversity is crucial to explore the search space effectively and avoid being stuck in local optimal. Conversely, when optimising the mean-objective, convergence is an important consideration to improve the average performance and achieve a balanced workload distribution.

E. Structure Analyses of Elements in Ensemble

Studying the structure of elements in an ensemble can help users understand the principles of each element. This understanding can give users more confidence in using the evolved ensemble [71]. In this case, we consider the tree structures of routing rules in an evolved ensemble of a single run on the scenario <Fmax,0.85> of EGP^e as an example. Fig. 16 illustrates the structures of routing rules, the training fitness of the ensemble, and the training ensemble contribution of each rule. The overall training fitness of this ensemble is 1359.25, better than the training fitness of each individual element, which ranges from 1424.19 to 1505.83. Additionally, the ensemble contribution of each individual element ranges from 0.92 to 0.99. The tree structures of these five routing rules can be expressed in the following simplified expressions:

$$\begin{aligned}
R_1 =& \min(\min(NIQ, SL) + 2PT + TRANT, WIQ) + PT + \\
& \min(\max(NOR - PT, 1), PT - NIQ + \max(PT, 1)) + \\
& TRANT + \max(TRANT, WIQ) \\
R_2 =& \min(\min(WIQ, PT - NOR + \min(SL, WKR)) + \\
& TRANT, WIQ) + PT + \min(\max(TRANT, WIQ), \\
& \frac{NOR - NIQ}{TRANT}) + \max(TRANT, WIQ) \\
R_3 =& 2\min(\min(NIQ, SL) + PT + TRANT, WIQ) + \\
& WIQ + TRANT \\
R_4 =& \min(\min(NIQ, SL) + PT + TRANT, WIQ) + PT + \\
& \min(\frac{\max(\min(WIQ, PT), \frac{NOR}{OWT})}{TRANT}, TIS - \frac{PT}{NIQ}) + \\
& \max(TRANT, WIQ) \\
R_5 =& \min(TIS, WIQ) + 2PT + TRANT - NIQ + \\
& 2\max(TRANT, WIQ) + \frac{NOR}{OWT}
\end{aligned} \tag{21}$$

To provide specific details, as illustrated in Eq. (21), the routing rule R_1 is composed of 6 terminals (NIQ, SL, PT, TRANT, WIQ, and NOR), with PT being the most frequently utilised terminal (appearing 6 times). Following closely is TRANT, employed 3 times. Meanwhile, the routing rule R_2 consists of 7 terminals (WIQ, PT, NOR, SL, WKR, TRANT, and NIQ), with WIQ and TRANT being the predominant terminals (each occurring 4 times). Additionally, PT and NOR are used 2 times. As for routing rule R_3, it involves 5 terminals (NIQ, SL, PT, TRANT, and WIQ). TRANT and WIQ are the terminals most frequently employed, each appearing 3 times, followed by NIQ, SL, and PT, each used 2 times. Moving on to routing rule R_4, it encompasses 8 terminals (NIQ, SL, PT, TRANT, WIQ, NOR, OWT, and TIS). PT claims the highest frequency (occurring 4 times). Both TRANT and WIQ are employed 3 times. Lastly, the routing rule R_5 is combined with 7 terminals (TIS, WIQ, PT, TRANT, NIQ, NOR, and OWT). Within this combination, WIQ and TRANT lead the pack with 3 appearances. PT follows with 2 appearances. These rules behave differently (give different training fitness), but there are some similarities. We can see that there are some terminals (NIQ, PT, TRANT, and WIQ) used by all these five routing rules, also PT and TRANT play important roles (high use frequency) in the routing decision point, which is expected, as processing time (PT) and transportation time (TRANT) are two important factors of the machine. Moreover, it is observed that some elements in the ensemble share the same subtrees. For example, R_1, R_2, R_4, and R_5 all have the subtree $\max(TRANT, WIQ)$. R_3 and R_4 both have the subtree $\min(\min(NIQ, SL) + PT + TRANT, WIQ)$.

We can see that, the routing rules with some general subtrees can cover most decision points, but some different subtrees are required for some specific decision points, which means these different subtrees can help these rules complement with each other to make better overall decisions. Such a phenomenon inspires us to aim for learning a good rule that is generalisable to most decision points while also having subtrees that can play crucial roles in specific decision points.

VIII. Conclusions

In this paper, we propose a novel EGP method named EGP^e, which allows the evolution of single individuals and ensembles together for solving the DFJSS problem. Extensive experiments and analyses demonstrate the effectiveness of our proposed method in terms of the evolved scheduling heuristic quality compared to existing recent popular GP methods. To be precise, the proposed method improves the test performance by 0.73% and saves the training time by 8.72% compared with the classic GP. Our proposed strategies, including ensemble construction and selection, and genetic operators considering both single individuals and ensembles, have also been verified to be effective in improving the performance of our EGP^e method. Further analyses show that these strategies can generate high-quality single individuals and ensembles by preserving population diversity and supporting high ensemble contributions from ensemble elements. Moreover, the structural analyses of elements in the ensemble demonstrate that a promising ensemble contains elements with both shared subtrees and distinctive subtrees, allowing for effective complementarity and finally leading to improved overall decision-making capabilities. This combination of the ensemble enables the ensemble to make reliable decisions for a wide range of decision points while also excelling in specific decision points. Overall, we believe that our findings contribute to the advancement of the field of GP and ensemble learning and have the potential for practical applications in real-world scheduling scenarios.

Further research could continue to improve the proposed EGP^e method by: 1) integrating surrogate models to accelerate the training process and enhance its efficacy, 2) extending the EGP^e method to address the multi-objective DFJSS problem, which considers multiple conflicting objectives simultaneously, and 3) developing a more generalised framework for a smooth application of the proposed method in solving a wider range of problems. Since the proposed method involves many components such as the new genetic operators and multi-case fitness evaluation, which make it complex.

References

[1] 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.

[2] M. Xu, Y. Mei, F. Zhang, and M. Zhang, "Genetic programming with diverse partner selection for dynamic flexible job shop scheduling," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2022, pp. 615–618.

[3] Y. Fu and P. W. Anderson, "Application of statistical mechanics to np-complete problems in combinatorial optimisation," *Journal of Physics A: Mathematical and General*, vol. 19, no. 9, p. 1605, 1986.

[4] X. Long, J. Zhang, K. Zhou, and T. Jin, "Dynamic self-learning artificial bee colony optimization algorithm for flexible job-shop scheduling problem with job insertion," *Processes*, vol. 10, no. 3, p. 571, 2022.

[5] Y. Gui, D. Tang, H. Zhu, Y. Zhang, and Z. Zhang, "Dynamic scheduling for flexible job shop using a deep reinforcement learning approach," *Computers & Industrial Engineering*, vol. 180, p. 109255, 2023, doi:10.1016/j.cie.2023.109255.

[6] 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, 2013.

[7] C. Destouet, H. Tlahig, B. Bettayeb, and B. Mazari, "Flexible job shop scheduling problem under industry 5.0: A survey on human reintegration, environmental consideration and resilience improvement," *Journal of Manufacturing Systems*, vol. 67, pp. 155–173, 2023.

[8] M. Thenarasu, K. Rameshkumar, J. Rousseau, and S. Anbuudayasankar, "Development and analysis of priority decision rules using mcdm approach for a flexible job shop scheduling: A simulation study," *Simulation Modelling Practice and Theory*, vol. 114, p. 102416, 2022.

[9] N. Zhu, G. Gong, D. Lu, D. Huang, N. Peng, and H. Qi, "An effective reformative memetic algorithm for distributed flexible job-shop scheduling problem with order cancellation," *Expert Systems with Applications*, vol. 237, p. 121205, 2024, doi:10.1016/j.eswa.2023.121205.

[10] C. Zhang, Y. Zhou, K. Peng, X. Li, K. Lian, and S. Zhang, "Dynamic flexible job shop scheduling method based on improved gene expression programming," *Measurement and Control*, vol. 54, no. 7-8, pp. 1136–1146, 2021.

[11] Y. Zhou, J. Yang, and L. 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.

[12] 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.

[13] 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.

[14] F. J. Gil-Gala, M. R. Sierra, C. Mencía, and R. Varela, "Surrogate model for memetic genetic programming with application to the one machine scheduling problem with time-varying capacity," *Expert Systems with Applications*, vol. 233, p. 120916, 2023.

[15] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Survey on genetic programming and machine learning techniques for heuristic design in job shop scheduling," *IEEE Transactions on Evolutionary Computation*, 2023, doi:10.1109/TEVC.2023.3255246.

[16] 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.

[17] A. Sitahong, Y. Yuan, M. Li, J. Ma, Z. Ba, and Y. Lu, "Designing dispatching rules via novel genetic programming with feature selection in dynamic job-shop scheduling," *Processes*, vol. 11, no. 1, p. 65, 2022.

[18] M. Xu, F. Zhang, Y. Mei, and M. Zhang, "Genetic programming with multi-case fitness for dynamic flexible job shop scheduling," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2022, pp. 1–8.

[19] ——, "Genetic programming with archive for dynamic flexible job shop scheduling," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2021, pp. 2117–2124.

[20] 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.

[21] M. Đurasević, L. Planinić, F. J. G. Gala, and D. Jakobović, "Novel ensemble collaboration method for dynamic scheduling problems," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2022, pp. 893–901.

[22] Z. S. Khozani, M. J. S. Safari, A. D. Mehr, and W. H. M. W. Mohtar, "An ensemble genetic programming approach to develop incipient sediment motion models in rectangular channels," *Journal of Hydrology*, vol. 584, p. 124753, 2020.

[23] T. G. Dietterich, "Ensemble methods in machine learning," in *Multiple Classifier Systems: First International Workshop*, 2000, pp. 1–15.

[24] K. Nag and N. R. Pal, "A multiobjective genetic programming-based ensemble for simultaneous feature selection and classification," *IEEE Transactions on Cybernetics*, vol. 46, no. 2, pp. 499–510, 2015.

[25] 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.

[26] Z. Wang, Z. Liang, R. Zeng, H. Yuan, and R. S. Srinivasan, "Identifying the optimal heterogeneous ensemble learning model for building energy prediction using the exhaustive search method," *Energy and Buildings*, vol. 281, p. 112763, 2023.

[27] H. Zhang, A. Zhou, Q. Chen, B. Xue, and M. Zhang, "Sr-forest: A genetic programming based heterogeneous ensemble learning method," *IEEE Transactions on Evolutionary Computation*, 2023, doi:10.1109/TEVC.2023.3243172.

[28] U. Bhowan, M. Johnston, M. Zhang, and X. Yao, "Evolving diverse ensembles using genetic programming for classification with unbalanced data," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 3, pp. 368–386, 2012.

[29] W. Smart and M. Zhang, "Using genetic programming for multiclass classification by simultaneously solving component binary classification problems," in *Proceedings of the European Conference on Genetic Programming*, 2005, pp. 227–239.

[30] M. Đurasević and D. Jakobović, "Comparison of ensemble learning methods for creating ensembles of dispatching rules for the unrelated machines environment," *Genetic Programming and Evolvable Machines*, vol. 19, pp. 53–92, 2018.

[31] 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*, vol. 21, no. 4, pp. 553–563, 2022.

[32] 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.

[33] F. J. Gil Gala and R. Varela, "Genetic algorithm to evolve ensembles of rules for on-line scheduling on single machine with variable capacity," in *Proceedings of the International Work-Conference on the Interplay Between Natural and Artificial Computation*, 2019, pp. 223–233.

[34] 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*, 2015, pp. 92–104.

[35] S. Wang, Y. Mei, and M. Zhang, "Novel ensemble genetic programming hyper-heuristics for uncertain capacitated arc routing problem," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019, pp. 1093–1101.

[36] N. M. Rodrigues, J. E. Batista, and S. Silva, "Ensemble genetic programming," in *Proceedings of the European Conference on Genetic Programming*, 2020, pp. 151–166.

[37] M. Xu, Y. Mei, F. Zhang, and M. Zhang, "Genetic programming with lexicase selection for large-scale dynamic flexible job shop scheduling," *IEEE Transactions on Evolutionary Computation*, 2023, doi:10.1109/TEVC.2023.3244607.

[38] H.-B. Song and J. Lin, "A genetic programming hyper-heuristic for the distributed assembly permutation flow-shop scheduling problem with sequence dependent setup times," *Swarm and Evolutionary Computation*, vol. 60, p. 100807, 2021.

[39] S. Shady, T. Kaihara, N. Fujii, and D. Kokuryo, "Feature selection approach for evolving reactive scheduling policies for dynamic job shop scheduling problem using gene expression programming," *International Journal of Production Research*, vol. 61, no. 15, pp. 5029–5052, 2023.

[40] N. Q. Uy, N. T. Hien, N. X. Hoai, and M. O'Neill, "Improving the generalisation ability of genetic programming with semantic similarity based crossover," in *Proceedings of the European Conference on Genetic Programming*, 2010, pp. 184–195.

[41] M. Xu, Y. Mei, F. Zhang, and M. Zhang, "Genetic programming with cluster selection for dynamic flexible job shop scheduling," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2022, pp. 1–8.

[42] H. Zhang, Q. Chen, A. Tonda, B. Xue, W. Banzhaf, and M. Zhang, "Map-elites with cosine-similarity for evolutionary ensemble learning," in *Proceedings of the European Conference on Genetic Programming*, 2023, pp. 84–100.

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

[44] 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.

[45] 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.

[46] 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 Genetic and Evolutionary Computation Conference Companion*, 2015, pp. 1451–1452.

[47] M. Đurasević and D. Jakobović, "Creating dispatching rules by simple ensemble combination," *Journal of Heuristics*, vol. 25, pp. 959–1013, 2019.

[48] 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*, 2016, pp. 115–132.

[49] M. Đumić and D. Jakobović, "Ensembles of priority rules for resource constrained project scheduling problem," *Applied Soft Computing*, vol. 110, p. 107606, 2021.

[50] 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 Genetic and Evolutionary Computation Conference Companion*, 2016, pp. 109–110.

[51] F. J. Gil Gala, M. Đurasević, R. Varela, and D. Jakobović, "Ensembles of priority rules to solve one machine scheduling problem in real-time," *Information Sciences*, 2023, doi:10.1016/j.ins.2023.03.114.

[52] 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.

[53] 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.

[54] J. Mohan, K. Lanka, and A. N. Rao, "A review of dynamic job shop scheduling techniques," *Procedia Manufacturing*, vol. 30, pp. 34–39, 2019.

[55] Q. Zhang, H. Manier, and M.-A. Manier, "A modified shifting bottleneck heuristic and disjunctive graph for job shop scheduling problems with transportation constraints," *International Journal of Production Research*, vol. 52, no. 4, pp. 985–1002, 2014.

[56] V. Vinod and R. Sridharan, "Dynamic job-shop scheduling with sequence-dependent setup times: simulation modeling and analysis," *The International Journal of Advanced Manufacturing Technology*, vol. 36, pp. 355–372, 2008.

[57] P. Sharma and A. Jain, "Stochastic dynamic job shop scheduling with sequence-dependent setup times: simulation experimentation," *Journal of Engineering and Technology*, vol. 5, no. 1, p. 19, 2015.

[58] K. C. W. Lim, L.-P. Wong, and J. F. Chin, "Hyper-heuristic for flexible job shop scheduling problem with stochastic job arrivals," *Manufacturing Letters*, vol. 36, pp. 5–8, 2023.

[59] W. Ren, Y. Yan, Y. Hu, and Y. Guan, "Joint optimisation for dynamic flexible job-shop scheduling problem with transportation time and resource constraints," *International Journal of Production Research*, vol. 60, no. 18, pp. 5675–5696, 2022.

[60] L. Wei, J. He, Z. Guo, and Z. Hu, "A multi-objective migrating birds optimization algorithm based on game theory for dynamic flexible job shop scheduling problem," *Expert Systems with Applications*, vol. 227, p. 120268, 2023, doi:10.1016/j.eswa.2023.120268.

[61] H. Wang, Y. Jiang, H. Wang, and H. Luo, "An online optimization scheme of the dynamic flexible job shop scheduling problem for intelligent manufacturing," in *Proceedings of the International Conference on Industrial Artificial Intelligence*, 2022, pp. 1–6.

[62] L. Meng, C. Zhang, Y. Ren, B. Zhang, and C. Lv, "Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem," *Computers & Industrial Engineering*, vol. 142, p. 106347, 2020.

[63] K. Lei, P. Guo, W. Zhao, Y. Wang, L. Qian, X. Meng, and L. Tang, "A multi-action deep reinforcement learning framework for flexible job-shop scheduling problem," *Expert Systems with Applications*, vol. 205, p. 117796, 2022, doi:10.1016/j.eswa.2022.117796.

[64] K. Lei, P. Guo, Y. Wang, J. Zhang, X. Meng, and L. Qian, "Large-scale dynamic scheduling for flexible job-shop with random arrivals of new jobs by hierarchical reinforcement learning," *IEEE Transactions on Industrial Informatics*, 2023, doi:10.1109/TII.2023.3272661.

[65] J. R. Koza *et al.*, *Genetic programming II*. MIT press Cambridge, 1994, vol. 17.

[66] 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*, 2018, pp. 472–484.

[67] J. E. Batista and S. Silva, "Comparative study of classifier performance using automatic feature construction by m3gp," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2022, pp. 1–8.

[68] M. Đurasević, F. J. G. Gala, D. Jakobović, and C. A. C. Coello, "Combining single objective dispatching rules into multi-objective ensembles for the dynamic unrelated machines environment," *Swarm and Evolutionary Computation*, p. 101318, 2023.

[69] D. W. Zimmerman and B. D. Zumbo, "Relative power of the wilcoxon test, the friedman test, and repeated-measures anova on ranks," *Journal of Experimental Education*, vol. 62, no. 1, pp. 75–86, 1993.

[70] G. Divine, H. J. Norton, R. Hunt, and J. Dienemann, "A review of analysis and sample size calculation considerations for wilcoxon tests," *Anesthesia & Analgesia*, vol. 117, no. 3, pp. 699–710, 2013.

[71] Y. Mei, Q. Chen, A. Lensen, B. Xue, and M. Zhang, "Explainable artificial intelligence by genetic programming: A survey," *IEEE Transactions on Evolutionary Computation*, 2022, doi:10.1109/TEVC.2022.3225509.