Expert Systems With Applications 322 (2026) 132158

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Genetic programming with advanced diverse partner selection for dynamic scheduling

Meng Xu [a,*], Yi Mei [b], Fangfang Zhang [b], Yew Soon Ong [c], Mengjie Zhang [b]

[a] Beijing Key Laboratory of Lightweight Intelligent System, Beijing Institute of Technology, 100081, Beijing, China
[b] School of Engineering and Computer Science, Evolutionary Computation Research Group, Victoria University of Wellington, 6140, Wellington, New Zealand
[c] College of Computing & Data Science, Nanyang Technological University, 639798, Singapore

ARTICLE INFO

Keywords:
Dynamic flexible job shop scheduling
Genetic programming
Parent selection
Brood recombination
Hyper-heuristic
Scheduling heuristic

ABSTRACT

Dynamic flexible job shop scheduling requires simultaneous optimisation of machine assignment and operation sequencing to generate effective schedules in uncertain environments. Genetic programming (GP) has shown promise in evolving scheduling heuristics for such problems. However, traditional GP parent selection methods, often based solely on fitness, might favor parents with similar behaviours, limiting the exploration of diverse solutions. Diverse partner selection (DPS) has been employed in GP to identify parents with complementary strengths for DFJSS, but the potential of DPS-selected parents has not been fully exploited. To address this gap, this paper proposes a novel GP approach, GPDPS^o, which integrates DPS with a new crossover mechanism. This crossover leverages brood recombination to generate a pool of candidate offspring, employs a surrogate model to efficiently estimate their performance, and incorporates an offspring selection strategy to retain the most promising individuals for the next generation. Experimental results demonstrate that GPDPS^o significantly outperforms traditional GP and other benchmark methods across various scheduling scenarios. This improvement is attributed to the synergistic integration of DPS, brood recombination, and surrogate-based offspring selection, which effectively exploits parent strengths, prioritizes offspring quality, and enables the evolution of more sophisticated scheduling heuristics.

1. Introduction

Flexible job shop scheduling (FJSS) is a classical optimisation problem in manufacturing that seeks the optimal sequence of operations on multiple machines to minimise objectives such as tardiness or flowtime (Li et al., 2023a). By allowing each operation to be processed on one of several eligible machines, FJSS introduces routing flexibility, which enhances system adaptability but also substantially increases the combinatorial complexity of optimisation (Zhang et al., 2023c). Building upon this, the *dynamic* FJSS (DFJSS) (Zhang et al., 2021d) further incorporates real-world uncertainties such as stochastic job arrivals and processing time variations. Among these, random job arrivals over time (Zhang et al., 2021f) are the most frequent dynamic events in manufacturing systems and constitute the primary focus of this study. Consequently, DFJSS requires the simultaneous optimisation of *machine assignment (routing)* and *operation sequencing* under continuously changing conditions. Each job comprises multiple operations that must be executed on specific machines following technological precedence constraints, rendering DFJSS a particularly challenging optimisation problem.

Two interdependent decisions must be made at each scheduling step: routing determines the machine to process a ready operation, while sequencing decides the next operation for an idle machine. Traditional optimisation approaches, including mathematical programming (Stecke, 1983) and metaheuristics such as tabu search (Saidi-Mehrabad & Fattahi, 2007; Vilcot & Billaut, 2011), artificial bee colony (Zhao et al., 2022b), and memetic algorithms (Li et al., 2024), have shown strong performance in static scheduling domains (Zhao et al., 2022a). However, these methods often struggle to handle the real-time decision-making requirements of DFJSS due to its dynamic and computationally intensive nature. To address this, *scheduling heuristics*, which dynamically assign priority values to candidate operations or machines, have become an effective alternative for achieving near-optimal solutions in real time (Nie et al., 2013). These heuristics guide decision-making at each event point based on system states such as machine workload, job waiting time, and operation processing time. Nevertheless, manually designing effective

* Corresponding author.
E-mail addresses: meng.xu.vuw@gmail.com (M. Xu), yi.mei@ecs.vuw.ac.nz (Y. Mei), fangfang.zhang@ecs.vuw.ac.nz (F. Zhang), mengjie.zhang@ecs.vuw.ac.nz (M. Zhang).

https://doi.org/10.1016/j.eswa.2026.132158
Received 12 November 2025; Received in revised form 2 March 2026; Accepted 18 March 2026
Available online 8 April 2026

heuristics is time-consuming and highly dependent on domain expertise, motivating the development of automated and adaptive heuristic learning approaches.

Recently, learning-based approaches have gained popularity for addressing DFJSS problems, with deep reinforcement learning (DRL) (Ngwu et al., 2025) and genetic programming (GP) (Koza et al., 1994) being the two most prominent paradigms. DRL leverages neural networks to learn adaptive scheduling policies directly from interactions with the environment, offering strong performance in dynamic contexts. However, DRL models often produce black-box policies that lack interpretability, limiting their acceptance and practical deployment in manufacturing. In contrast, GP evolves symbolic, rule-based heuristics that are transparent, interpretable, and adaptable across problem settings (Miyashita, 2000; Nguyen et al., 2019b; Tay & Ho, 2008). Given the importance of interpretability and efficiency for real-world DFJSS, this study focuses on advancing GP as the primary learning-based approach.

GP employs an iterative process of initialising individuals (heuristics), evaluating individuals, selecting parents, and applying crossover, mutation, and reproduction operators to generate new individuals (Tay & Ho, 2008). Parent selection plays an important role in GP, as it determines which individuals contribute their genetic material to the next generation (Helmuth & Abdelhady, 2020). Classical selection methods such as tournament selection, fitness-proportional selection, and rank selection (Jebari et al., 2013; Kotanchek et al., 2007; Xie, 2009) predominantly depend on fitness (Fang & Li, 2010), which can sometimes result in the selection of parents with similar behaviours. This occurs because individuals with better fitness scores are more likely to exhibit comparable characteristics, limiting the exploration of diverse solutions. To address this, advanced parent selection methods such as batch tournament selection (De Melo et al., 2019) and lexicase selection (Metevier et al., 2019) introduce diversity by sampling different subsets of training cases.

Crossover is a crucial operator in GP, responsible for combining genetic material from selected parents to create offspring. Ideally, selected parents should possess not only high quality but also complementary strengths. Various selection methods with guiding strategies have been proposed to achieve this. Comparative partner selection (CPS) (Day & Nandi, 2008) aims to choose parental pairs based on their strengths in different training cases but may overlook potentially valuable parent combinations (Aslam et al., 2018). This limitation motivated the development of diverse partner selection (DPS) (Aslam et al., 2018), which has shown promise in identifying high-quality and complementary parent combinations. DPS has been adapted for DFJSS through a multi-case fitness evaluation strategy and a new benefit function (Xu et al., 2022a), demonstrating good performance across different DFJSS scenarios. Despite these advances, existing GP approaches for DFJSS, including DPS-based methods, share three critical limitations that this paper addresses:

- Limited exploration of genetic combinations: While DPS effectively identifies complementary parent pairs, it subsequently employs standard crossover, generating only one or two offspring per parent pair. This fails to fully exploit the potential of the identified complementary pairs, as many promising genetic combinations remain unexplored. Brood recombination (Tackett & Carmi, 1994; Zhang et al., 2022a) offers a solution by generating multiple offspring from the same parent pair through multiple crossover operations with varying crossover points. However, brood recombination has not been widely adopted in large-scale DFJSS, as evaluating the numerous offspring using true simulation is computationally prohibitive.
- Absence of diversity-aware offspring selection: Existing methods select offspring for the next generation based mainly on fitness (or multi-case fitness). This fitness-only selection risks losing phenotypically diverse individuals that may carry valuable building blocks for future generations, even when their immediate fitness is not optimal. The relationship between phenotypic behaviour and long-term evolutionary potential is not captured by fitness alone.
- Disjointed integration of components: Prior work has treated parent selection, crossover, and offspring selection as largely independent modules. What is missing is a *unified framework* where these components are designed to work synergistically: parent selection identifies complementary pairs, crossover thoroughly explores their genetic combinations, and offspring selection preserves both quality and diversity. Without such integration, the full potential of each component remains unrealised.

To overcome these limitations, this paper proposes GPDPS^o, a novel GP framework that introduces three key innovations:

- Brood recombination with surrogate-based evaluation: GPDPS^o integrates brood recombination into GP for large-scale DFJSS. To overcome the computational barrier, we employ a K-nearest neighbours (KNN) surrogate model that estimates multi-case fitness using phenotypic characteristics (PC), a vector capturing each heuristic's routing and sequencing decisions at representative decision points. This enables the algorithm to generate and screen up to $2 \times x$ offspring per parent pair (where x is the number of brood crossovers) while maintaining the same true evaluation budget as standard GP. The surrogate is not a peripheral efficiency trick; it fundamentally transforms brood recombination from a computationally prohibitive strategy into a practical and effective one, enabling exploration of a vastly larger space of genetic combinations.
- PC-aware offspring selection strategy: We propose a novel two-stage selection mechanism that explicitly considers both phenotypic diversity and performance. Offspring are first filtered based on PC similarity, retaining only the best-performing individual among those with identical behavioural characteristics. The filtered offspring are then prioritised: those with PCs distinct from any previously accepted offspring form a high-priority group, ensuring that diverse building blocks enter the next generation. Only within these priority groups does fitness determine final selection. This explicitly addresses the diversity loss inherent in fitness-only selection and maintains a richer pool of genetic material throughout evolution.
- Synergistic integration of components: The true novelty of GPDPS^o lies in how these components work together in an integrated framework. DPS identifies parent pairs that are both high-quality and complementary. Brood recombination, enabled by the surrogate, thoroughly explores the space of genetic combinations from these promising pairs. The PC-aware selection then ensures that the offspring entering the next generation are both high-performing and phenotypically diverse, which in turn provides richer material for future DPS selections. This creates a positive feedback loop where diversity leads to more effective exploitation, and exploitation informs further exploration, a dynamic that no existing GP method for DFJSS achieves.

The remainder of this paper is organised as follows. Section 2 introduces the background and related work. Section 3 provides detailed descriptions of the proposed GPDPS^o framework. The experimental design is presented in Section 4, followed by the experimental results in Section 5. Further analysis is conducted in Section 6, and the paper concludes with Section 7.

2. Background

2.1. Dynamic flexible job shop scheduling

The shop floor consists of a set of heterogeneous machines, denoted by $\mathcal{M} = \{M_1, ..., M_p, ..., M_m\}$, with varying processing rates and locations (Li et al., 2023b). Each machine M_p has a processing rate of $\gamma(M_p)$. Jobs arrive at the shop floor over time and are represented by the set $\mathcal{J} = \{J_1, J_2, ..., J_n\}$. Each job J_i is characterised by:

- Release time (r_j): The earliest time at which the job can begin processing.
- Due date (d_j): The latest time by which the job is expected to be completed.
- Weight (w_j): A numerical value representing the job's importance.
- Operation sequence ($\mathcal{O}_i = \{O_{i1}, O_{i2}, ..., O_{ik}\}$): An ordered list of operations that the job requires to be completed.

Each operation O_{ij} within a job's sequence has a work of l_{ij} and can be processed by one of its compatible machines from the set $\pi(O_{ij}) \subseteq \mathcal{M}$. The processing time of an operation O_{ij} on a machine M_p (where $M_p \in \pi(O_{ij})$) is denoted by $\sigma_{M_p}(O_{ij}) = l_{ij}/\gamma(M_p)$ and is calculated as the work l_{ij} divided by the processing rate of the machine, $\gamma(M_p)$. Moreover, once the routing decision assigns a machine (M_q) to a ready operation, the operation needs to be transported from its current location (L_p) to the selected machine's location (L_q). This transportation incurs a time cost, denoted by $T_{p,q} = dis_{p,q}/v$, which depends on two factors: 1) the physical distance $dis_{p,q}$ between the locations of machines L_p and L_q; 2) the transport speed (v). The constraints and assumptions of the DFJSS problems are shown as follows.

- The sequence of operations for each job is predetermined upon arrival at the shop floor. An operation can only be processed on its selected machine after all its preceding operations are completed.
- Once an operation begins processing on a machine, it cannot be interrupted or suspended until completion.
- Each operation can only be processed by one of its compatible machines.
- Each machine can handle a maximum of one operation at a time.
- The processing rate and location of each machine remain constant throughout the scheduling process.

This paper investigates three objectives for solving the DFJSS problems: minimising max-flowtime ($Fmax$), minimising mean-flowtime ($Fmean$), and minimising max-weighted-tardiness ($WTmax$). These three objectives are formulated as:

- Minimising $Fmax = \min_{S \in \mathcal{S}} \max_{j \in \{1,2,\ldots,n\}} \left\{C_j(S) - r_j\right\}$
- Minimising $Fmean = \min_{S \in \mathcal{S}} \frac{1}{n} \sum_{j=1}^{n} \left(C_j(S) - r_j\right)$
- Minimising $WTmax = \min_{S \in \mathcal{S}} \max_{j \in \{1,2,\ldots,n\}} \max\left\{w_j(C_j(S) - d_j), 0\right\}$

where n represents the number of jobs. S represents a valid schedule in the solution space $\mathcal{S}$. $C_j(S)$ denotes the completion time of job J_i under schedule S.

2.2. Related work

2.2.1. Learning-based approaches for DFJSS

Both DRL and GP have emerged as popular learning-based approaches for tackling DFJSS problems (Xu et al., 2025). These methods share a common goal: to learn adaptive scheduling policies that can make rapid decisions under dynamic and uncertain conditions. However, they differ fundamentally in how the policies are represented, trained, and applied (Xu et al., 2025).

DRL methods, by leveraging neural networks, learn adaptive scheduling policies through direct interactions with the scheduling environment, enabling fast decision-making under uncertainty (Xu et al., 2025). For instance, policy gradient and deep Q-learning approaches have been applied to dynamically assign machines and sequence operations on complex shop floors, demonstrating strong performance in handling job arrivals and other dynamic events (Ngwu et al., 2025; Zhang & Zhu, 2025). Actor-critic frameworks further improve stability and generalization by integrating value estimation with policy learning (Si et al., 2024). Despite these advantages, DRL approaches typically learn black-box policies that are difficult to interpret, which may limit their practical adoption in real-world manufacturing systems (Xu et al., 2025). These challenges motivate the exploration of GP as an alternative approach that emphasizes interpretability, efficiency, and adaptability.

GP provides an interpretable hyper-heuristic approach that evolves explicit scheduling heuristics (policies) rather than training black-box policies (Mei et al., 2022; Xu et al., 2024). GP operates in the heuristic space, automatically generating dispatching rules that generalize across different problem instances (Branke et al., 2015; Koza & Koza, 1992; Nguyen et al., 2019a; Zhang et al., 2021g). Unlike RL, which relies on neural approximations, GP evolves symbolic expressions that can be directly analyzed and understood by practitioners, offering transparency in decision-making (Xu et al., 2023). GP employs a two-phase process: training and test (Zhang et al., 2021a). Fig. 1 gives the training process of the classical GP method. During training, GP iteratively evolves a population of candidate scheduling heuristics. This process leverages a set of DFJSS instances to guide the evolution through a process of selection, crossover, mutation, and reproduction. The goal is to optimise the population towards heuristics that generate high-quality schedules for the training instances. Ultimately, the training phase outputs a single, well-performing scheduling heuristic. The evolved heuristic from the training phase is then evaluated on unseen DFJSS instances to assess its generalisability and effectiveness in solving problems beyond the training data. An illustrative example of the evolved scheduling heuristic (multi-tree-based representation) for DFJSS is depicted in Fig. 1, where PT, TIS, WIQ, and MRT are problem-specific features. PT denotes the processing time of an operation. TIS represents the time in the system of a job. WIQ is the workload of a machine, representing the total processing time of operations in the queue. MRT means the machine ready time. If a smaller value represents a higher priority, the routing rule in Fig. 1 prefers machines providing shorter processing time, smaller workloads, and larger ready time. The sequencing rule in Fig. 1 prefers operations with shorter processing times and shorter stay times in the system.

GP has been extensively explored for tackling DFJSS problems. Early advancements involved using multi-tree GP to concurrently evolve both routing and sequencing rules (Zhang et al., 2018a). Subsequent studies have focused on integrating machine learning techniques with GP to enhance its performance (Zhang et al., 2024). Techniques like those presented in Zakaria et al. (2021), Zhang et al. (2021c,d) employ feature selection to identify and eliminate irrelevant scheduling information from the problem space. This reduces the search space for GP, leading to faster training and potentially better heuristics learning. Studies like (Zhang et al., 2018b; Zhou et al., 2020) investigate using surrogate models to approximate the fitness evaluation process in GP. Surrogate models are faster to evaluate than actual scheduling simulations, allowing for significant efficiency gains during training without compromising the final heuristic's quality. Research in Zhang et al. (2023a) proposes a multi-task GP approach that leverages task relatedness to solve problems with multiple objectives simultaneously. This method adaptively selects assisted tasks based on the relatedness measure to solve the DFJSS with multiple tasks simultaneously. Surrogate models can also be employed within this framework for further efficiency improvements and knowledge transfer between tasks (Zhang et al., 2021e,f). Beyond the techniques mentioned above, GP variants have been also applied to address DFJSS problems with multiple objectives (Chen et al., 2018; Zhang et al., 2022b, 2023b).

2.2.2. Parent selection in GP

Selecting suitable parents plays a crucial role in the effectiveness of GP. Some main classical selection methods are commonly used to select candidates for offspring production in GP (Xie, 2009): tournament selection, fitness-proportional selection, rank selection, and Pareto tournament selection (Jebari et al., 2013; Kotanchek et al., 2007). Tournament selection (Fang & Li, 2010) stands as the most commonly used parent selection method in GP. It involves randomly sampling a set of candidate individuals and then selecting the individual with the best fitness as the parent. In fitness-proportional selection, individuals are selected

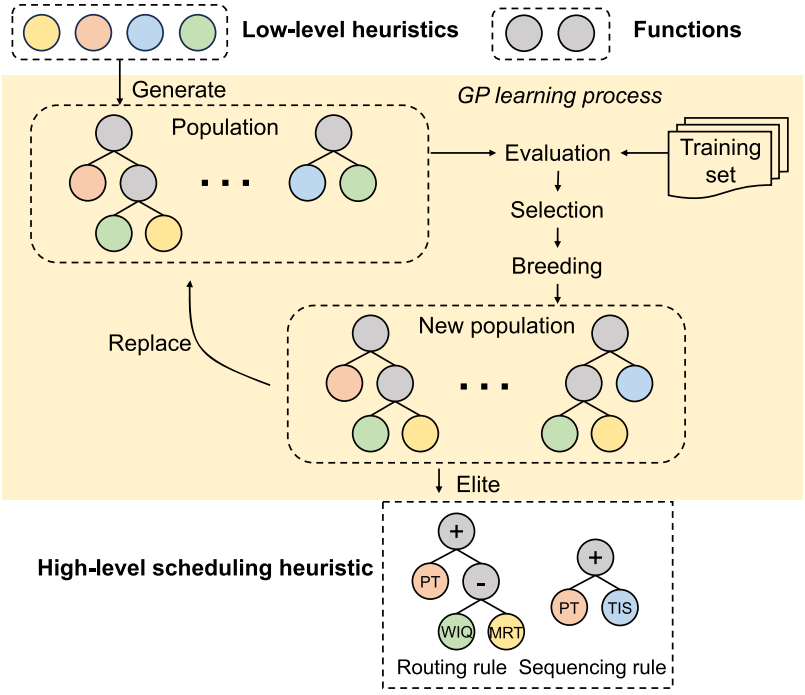

Fig. 1. The learning process of the classical GP method and an example of the learned scheduling heuristic.

based on their proportionate share of the fitness (Jebari et al., 2013). In rank selection, individuals are ranked based on their fitness values, and selection probabilities are assigned according to their rank rather than absolute fitness values (Xie, 2009). Pareto tournament selection is particularly used in multi-objective optimisation. It randomly samples a set of candidate individuals and selects those on the Pareto front as parents (Kotanchek et al., 2007). These selection methods can be applied to crossover, mutation, and reproduction. While mutation and reproduction only require a single parent to generate offspring, crossover necessitates the collaboration of two parents. However, when using these selection methods to choose the two parents for crossover, they are selected independently, potentially lacking effective collaboration to produce high-quality offspring. Lexicase selection (Metevier et al., 2019) is designed to select parents based on different orders of training cases. Studies have demonstrated that lexicase selection can yield more diverse parents and achieve superior performance compared to tournament selection (Aenugu & Spector, 2019; La Cava et al., 2016; Spector et al., 2018; Xu et al., 2023). Nevertheless, neither tournament selection nor lexicase selection incorporates a mechanism to ensure that the selected parents are a good match. Consequently, the two parents chosen for crossover might not necessarily offer a positive influence on each other.

CPS (Day & Nandi, 2008) and DPS (Aslam et al., 2018) are designed to select a pair of complementary parents. CPS seeks to identify parental pairs with strengths across different training cases. However, CPS can overlook certain high-quality parental pairs. DPS, on the other hand, offers a superior strategy by addressing the limitations of CPS (Aslam et al., 2018). In traditional DPS, the selection process begins with two individuals chosen via tournament selection. Each individual's behaviour is characterised by a binary string (Day & Nandi, 2008), reflecting its effectiveness across various cases. To adapt DPS for DFJSS, a novel strategy to assess whether one individual can complement the other is proposed based on multi-case fitness evaluation (Xu et al., 2022a). DPS has demonstrated improved performance on various combinatorial optimisation problems (Aslam et al., 2018), including DFJSS (Xu et al., 2022a). However, the effectiveness of the broader range of genetic combinations from parents selected by DPS for DFJSS has not yet been explored. Therefore, a new strategy is needed to fully harness the potential of parents chosen by DPS for DFJSS.

3. New method

3.1. Overall framework

The overall framework of the proposed GPDPS^o for DFJSS is shown in Fig. 2. Initially, a population of individuals is created using the ramped-half-and-half method. In each generation, individuals undergo evaluation through a multi-case fitness assessment, which produces a list of case-fitness values. Following the multi-case fitness evaluation, the case-fitness values are normalised. These values determine the potential benefit of one individual to another during crossover in DPS. A number of elites, with the highest mean multi-case fitness, are then directly advanced to the next generation. Subsequently, the breeding process generates offspring through parent selection and genetic operators. Parents for mutation and reproduction are chosen using the traditional tournament selection, while parents for crossover are selected based on the normalised case-fitness values from the multi-case fitness evaluation. In the proposed GPDPS^o for DFJSS, we adopt the multi-tree individual representation and multi-tree-based reproduction and mutation, which have been used and described in Zhang et al. (2018a). Briefly speaking, each individual consists of two trees, one representing the routing rule, and the other representing the sequencing rule. Given a parent for reproduction, it is directly passed on to the next generation. In the case of mutation, a random tree is chosen from the parent, and a random subtree within that tree is replaced with a newly generated subtree. Further details on this process can be found in the original literature (Zhang et al., 2018a). For crossover, this paper proposes a novel crossover strategy incorporating the brood recombination technique. It aims to leverage the strengths of the parent pairs identified by DPS. The crossover process utilises brood recombination, generating a number of candidate offspring. Only the two top candidates are then selected for the next generation. During the crossover of two parents (Zhang et al., 2018a), a random tree is selected from one parent. Subtrees within this chosen tree are then swapped with the corresponding subtrees in the

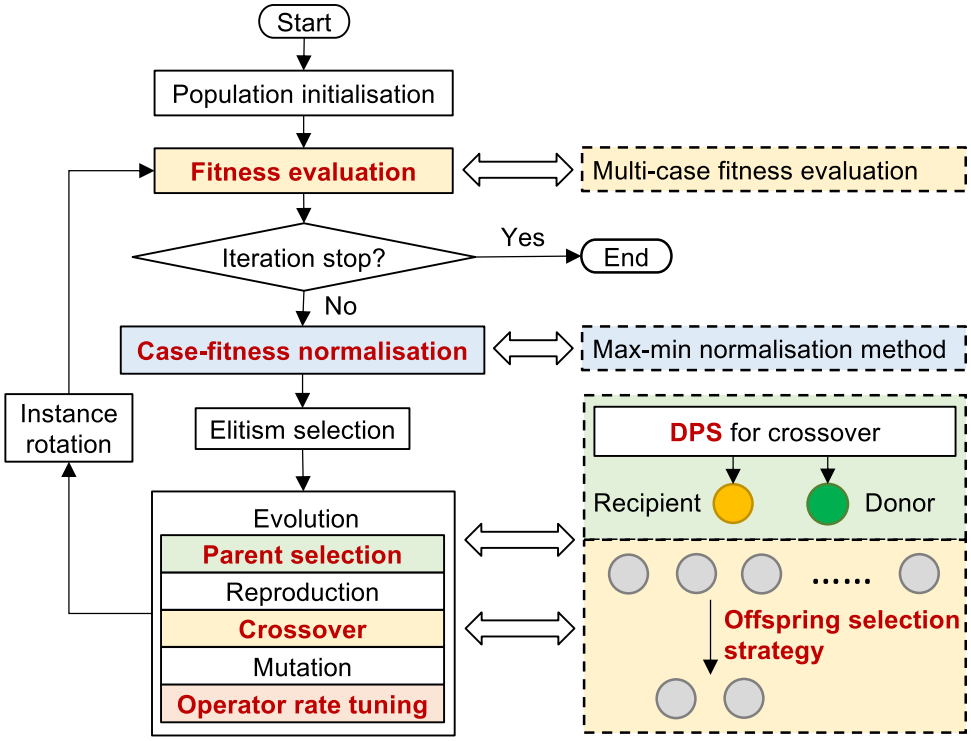

Fig. 2. The flowchart of the GPDPS^o method for DFJSS.

other parent's tree. And the other tree is directly swapped. To efficiently assess the multi-case fitness of the generated offspring without increasing computational time, a KNN surrogate model is employed. A separate offspring selection strategy is proposed to identify the two top candidates from the generated candidates based on their estimated multi-case fitness.

Throughout the breeding process, crossover and mutation rates are dynamically adjusted based on the frequency of failures in selecting a suitable second parent through DPS. This operator rate tuning strategy, derived from the original method (Aslam et al., 2018), aims to balance exploration and exploitation. When the DPS algorithm fails to find an appropriate second parent, the crossover rate is decreased, and the mutation rate is correspondingly increased by $1/popsize$. This adjustment promotes broader exploration of the search space when parent selection is challenging. After the breeding process, the crossover and mutation rates are reset to their default values for the next generation. These steps are repeated until a stopping criterion is met, at which point the best individual is returned.

Overall, GPDPS^o introduces several key enhancements over the conventional GP method (Zhang et al., 2018a), with the primary goal of improving the selection and breeding process to generate more effective scheduling heuristics. The main differences can be summarized as follows: (1) multi-case fitness evaluation, (2) case-fitness normalization, (3) DPS-based parent selection for crossover, (4) crossover using brood recombination combined with surrogate-driven fitness estimation and offspring selection, and (5) adaptive operator rate tuning. Among these, the most significant advancement of GPDPS^o from the baseline GPDPS (Xu et al., 2022a) lies in step (4) crossover and fitness estimation. While DPS encourages the pairing of parents that are both high-quality and complementary, GPDPS^o further strengthens this process through a novel crossover mechanism that integrates brood recombination, surrogate-based offspring evaluation, and a targeted offspring selection strategy. Collectively, these components, when coupled with DPS, form the GPDPS^o framework, which achieves an effective balance between solution diversity, solution quality, and computational efficiency. A detailed explanation of each modification is provided in the following sections.

3.2. Multi-case fitness evaluation

Multi-case fitness evaluation is proposed for DFJSS to get the standard fitness and a list of case-fitnesses instead of only the standard fitness (Xu et al., 2023). This paper proposes using this evaluation strategy for two purposes. First, it calculates a list of case-fitness values to characterise the behaviour of the individual in different cases. Such a list will be used to select parents for crossover. Second, it calculates the mean fitness of the list of case-fitnesses of the individual (Xu et al., 2022b), which will be used for elitism selection and select parents for mutation and reproduction. Briefly speaking, without loss of generality, the multi-case fitness evaluation strategy divides the n jobs completed during the DFJSS simulation into c cases (groups), each containing g jobs. The objective value for each group of jobs is then treated as a case. Specifically, if the group size is 1, every single job is considered a case. If the group size equals to the number of jobs in the simulation, then the entire simulation is treated as a single case, and the multi-case fitness evaluation reduces to the standard fitness evaluation. For instance, if the objective is the mean flowtime, the case-fitness ($\mathbf{cf}_i(ind)$) of case i by the individual ind is calculated using Eq. (1).

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

If the objective is the max tardiness, the case-fitness $\mathbf{cf}_i(ind)$ of case i by the individual ind is calculated by Eq. (2).

$$\mathbf{cf}_i(ind) = \max_{j\in\{g\times(i-1)+1,\ldots,g\times i\}} T_j \tag{2}$$

The mean fitness can be calculated from the case-fitnesses. For both max-objectives and mean-objectives, the mean fitness can be calculated

by Eq. (3).

$$sf_{mean}(ind) = \frac{1}{c}\sum_{i=1}^{c} \mathbf{cf}_i \tag{3}$$

To avoid bias in any case during DPS, for each case, we normalise the case-fitness of all individuals (*pop*) in the population (Xu et al., 2022a). Specifically, for each case $i = 1, \ldots, c$, the normalisation is done by

$$\mathbf{cf}_i(ind) = \frac{\mathbf{cf}_i(ind) - \min\{\mathbf{cf}_i(ind^s) | ind^s \in pop\}}{\max\{\mathbf{cf}_i(ind^b) | ind^b \in pop\} - \min\{\mathbf{cf}_i(ind^s) | ind^s \in pop\}} \tag{4}$$

3.3. Diverse partner selection

Algorithm 1 outlines the DPS process used to identify parent individuals for crossover in GPDPS^o. The choice of DPS for DFJSS is motivated by two key characteristics of the problem domain. First, temporal heterogeneity of decision quality: a heuristic's performance is not uniform across all decision points. A given scheduling heuristic may perform excellently under high machine utilisation but poorly during low utilisation periods, or vice versa. This means that two heuristics with similar average fitness may have complementary strengths across different shop floor conditions. Second, DPS addresses the limitations of fitness-based parent selection more effectively than other diversity-preserving techniques. Rather than merely maintaining population diversity, DPS explicitly seeks behavioural complementarity by leveraging the fine-grained case-wise performance profiles provided by multi-case fitness evaluation. This allows DPS to identify parent pairs whose strengths and weaknesses are genuinely complementary, rather than simply selecting individuals with diverse phenotypes or genetic compositions.

3.3.1. Recipient selection

The algorithm begins by selecting the first parent, also called the "recipient", using tournament selection.

3.3.2. Iterative donor selection

A second parent, the "donor" candidate, is first chosen via tournament selection. Then, to assess the potential benefit of this donor candidate for the recipient, the algorithm calculates the actual influence (α) using the benefit function in Eq. (5):

$$\alpha(Ind_1, Ind_2) = \sum_{i=1}^{c} \frac{\mathbf{cf}_i(Ind_1) - \mathbf{cf}_i(Ind_2)}{\max(\mathbf{cf}_i(Ind_1), \mathbf{cf}_i(Ind_2))} \tag{5}$$

For each case i, the term $\frac{\mathbf{cf}_i(Ind_1) - \mathbf{cf}_i(Ind_2)}{\max(\mathbf{cf}_i(Ind_1), \mathbf{cf}_i(Ind_2))}$ is:

- *Positive* when $\mathbf{cf}_i(Ind_2) < \mathbf{cf}_i(Ind_1)$ (donor outperforms recipient on case i);
- *Negative* when $\mathbf{cf}_i(Ind_2) > \mathbf{cf}_i(Ind_1)$ (recipient outperforms donor on case i);
- *Zero* when $\mathbf{cf}_i(Ind_2) = \mathbf{cf}_i(Ind_1)$.

Algorithm 1: Diverse partner selection for DFJSS.

Input: The population: *pop*; Number of trials K; Benefit threshold β.
Output: Selected parents: Ind_1, Ind_2.

```
Select Ind_1 ∈ pop by tournament selection;
for t = 1 → K do
    Select Ind_2 ∈ pop by tournament selection;
    Calculate α(Ind_1, Ind_2) by Eq. (5);
    if α(Ind_1, Ind_2) > β then
        return Ind_1, Ind_2;
    end
end
Decrease (increase) crossover (mutation) rate by δ;
return Ind_1, Ind_2;
```

Therefore, a larger (more positive) value of α indicates that the donor outperforms the recipient across more cases, signifying greater potential benefit from combining the two parents. The summation across all cases aggregates these case-level advantages into a single measure of overall positive influence.

Expected Positive Influence (β): This parameter defines the minimum threshold for acceptable positive influence. With $\beta \geq 0$, the condition $\alpha > \beta$ ensures that the selected donor provides a net positive contribution across cases:

- If $\alpha(Ind_1, Ind_2) > \beta$, the chosen donor is confirmed as a suitable partner and selected;
- If $\alpha(Ind_1, Ind_2) \leq \beta$, the selection process continues. A new donor candidate is chosen via tournament selection, and its influence on the recipient is evaluated;
- A threshold value, denoted by K, limits the number of attempts to find a suitable donor. If no qualified donor is found within K attempts, the algorithm selects the last considered donor as the partner.

Illustrative example: Fig. 3 visualises an example of DPS donor selection. The left side shows the normalised case-fitness lists of three individuals, where Ind_1 represents the recipient, and Ind_2 and Ind_3 are candidate donors. Ind_2 exhibits a complementary performance pattern: it performs well in some cases where Ind_1 is weak (e.g., case 5), but poorly in others where Ind_1 is strong (e.g., case 8). In contrast, Ind_3 consistently outperforms Ind_1 across most cases, demonstrating uniformly better values. The right-hand side presents the computed α values. With a threshold of $\beta = 0$, $\alpha(Ind_1, Ind_3) = 1.857$ is positive and large, indicating that Ind_3 outperforms Ind_1 in the majority of cases, making it a highly beneficial donor. Conversely, $\alpha(Ind_1, Ind_2) = -0.98$ is negative, indicating that Ind_1 outperforms Ind_2 in more cases than the reverse. Although Ind_2 offers complementary strengths in specific cases (e.g., case 5), its overall net influence is negative. Thus, Ind_3 ($\alpha = 1.857 > \beta$) is selected as the donor, while Ind_2 ($\alpha = -0.98 < \beta$) is rejected. This example illustrates that DPS selects donors based on the aggregate positive influence across all cases, rather than relying solely on the presence of complementarity in isolated instances.

3.3.3. Operator rate tuning

The operator rate tuning strategy aims to maintain a balance between exploration (searching for new solutions) and exploitation (focusing on promising areas of the search space). The ability to find a suitable donor within the K attempts serves as an indicator of population diversity.

- Easy Donor Selection (within K attempts): If a qualified donor is identified within K attempts, it suggests the population is sufficiently diverse. No operator rate adjustment is necessary.
- Difficult Donor Selection (exceeds K attempts): If no qualified donor is found after K attempts, it implies the population may lack diversity. To address this, the algorithm reduces the crossover rate and increases the mutation rate by δ, encouraging broader exploration. The reproduction rate remains unchanged.

The operator rate tuning occurs within each generation, but at the beginning of every new generation, crossover and mutation rates are reset to their initial values. This ensures a consistent balance between exploration and exploitation throughout the evolutionary process.

3.4. Crossover with brood recombination

Once the recipient and donor parents are chosen using the DPS method, GPDPS^o capitalises on their strengths by employing brood recombination (Tackett, 1994). Brood recombination allows the recipient and donor to undergo crossover multiple times (denoted by x). This generates a larger number of offspring ($2 \times x$) compared to traditional

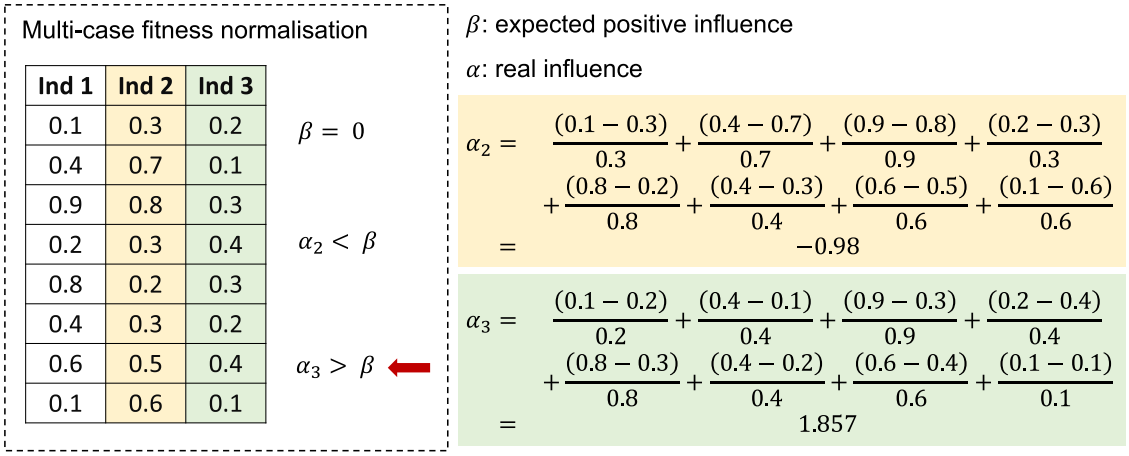

Fig. 3. Illustrative example of DPS donor selection. The recipient (Ind_1) has case-fitness vector $[0.1, 0.4, 0.9, 0.2, 0.8, 0.4, 0.6, 0.1]$. Candidate Ind_2 ($[0.3, 0.7, 0.8, 0.3, 0.2, 0.3, 0.5, 0.6]$) shows a mixed pattern but yields $\alpha = -0.98$, indicating net negative influence, and is rejected. Candidate Ind_3 ($[0.2, 0.1, 0.3, 0.4, 0.3, 0.2, 0.4, 0.1]$) consistently outperforms Ind_1, yielding $\alpha = 1.857 > \beta$, and is selected as the donor.

crossover. To promote diversity within the generated offspring, each crossover utilises randomly chosen and distinct crossover points within the parent trees. This helps explore a wider range of potential combinations between the parents' information. While brood recombination generates a larger pool of offspring, GPDPS^o employs a surrogate model for efficient offspring evaluation (details provided in Section 3.5) and implements a novel offspring selection strategy (details provided in Section 3.6) to choose only the two most promising individuals for the next generation.

3.5. Surrogate model for efficient offspring evaluation

Evaluating the fitness of individuals (scheduling heuristics) in DFJSS can be computationally expensive. To efficiently estimate the multi-case fitness of generated offspring during the breeding phase, GPDPS^o employs a KNN surrogate model with a single nearest neighbour, based on PC (Hildebrandt & Branke, 2015). PC is a vector of ranking numbers that captures an individual's decisions at a set of decision points, relative to those made by reference rules on the same decision situations in a DFJSS instance. We adopt a single nearest neighbour for several reasons. First, as a lazy learner, KNN requires no explicit training phase, it simply stores PC-fitness pairs and retrieves them when needed. Second, prior studies in surrogate-assisted GP for scheduling have demonstrated that retaining only the single best neighbour strikes an effective balance between estimation accuracy and computational efficiency (Zhang et al., 2021b,e; Zhu et al., 2023). Third, the intuitive interpretation, that an offspring's fitness is similar to that of its most behaviourally similar known individual, aligns naturally with the PC representation.

3.5.1. Construction of PCs

For each generation, the PC representation is constructed dynamically using the best individual from the current population. Specifically, the best heuristic is executed on a DFJSS instance, and its ranking decisions are recorded at all routing and sequencing decision points where there are seven candidates (i.e., seven operations or seven machines to choose from). From these, 20 routing decision points and 20 sequencing decision points are randomly selected, forming a 40-dimensional PC vector that characterises the heuristic's behaviour in that generation. This process is repeated once per generation, ensuring that the PC space remains aligned with the evolving search focus. The same PC definition, a 40-dimensional vector of ranking decisions, is applied consistently across all scenarios, although the specific decision instances naturally vary with problem characteristics and random seeds.

3.5.2. Surrogate-based fitness estimation

The KNN surrogate model calculates the similarity between each offspring's PC and the PCs of existing individuals in the population that have already undergone true evaluation. The similarity between PCs is computed using Euclidean distance, as illustrated in Eq. (6). A smaller Euclidean distance between individuals indicates a higher degree of phenotypic similarity.

$$\text{similarity} = \sum_{i=1}^{40} (d_i^{Ind_a} - d_i^{Ind_b})^2 \tag{6}$$

where i indexes the 40 decision points, and $d_i^{Ind_a}$ and $d_i^{Ind_b}$ represent the ranking decisions made by individuals Ind_a and Ind_b at decision point i, respectively.

3.5.3. Updating the surrogate

The KNN model requires no explicit training. Instead, it maintains a database of PC-fitness pairs from all individuals that have undergone true evaluation. This database is initialised with the first generation (all individuals truly evaluated) and is updated every generation as newly selected offspring undergo true evaluation before entering the population. Thus, the database continuously grows and becomes more representative of the search space over time.

3.5.4. True evaluation and surrogate correction

It is important to emphasise that surrogate-based estimation is used only during the offspring generation and selection phase to rapidly screen promising candidates. All individuals selected to form the next generation undergo true fitness evaluation on the DFJSS instance before being inserted into the population. This design ensures two critical properties:

- The population always contains individuals with verified true fitness, preventing error accumulation over generations.
- The PC-fitness database is continuously corrected with ground-truth values, maintaining the reliability of future surrogate estimations.

With respect to sensitivity to surrogate prediction errors, the consequences of an incorrect estimation are limited. An overestimated offspring undergoes true evaluation prior to entering the population, thereby revealing its true fitness. An underestimated offspring incurs only an opportunity cost and does not compromise population integrity. Because the surrogate influences only the set of candidates considered for true evaluation, and never dictates final population composition, the method remains inherently robust to prediction inaccuracies.

Table 1
An example of candidate offspring evaluation based on the *KNN* surrogate model.

Specie	ID	PC	similarity	fitness
Population	1	[1,3,4,2,7,3]	-	[20,13,27,24,15]
	2	[2,3,1,4,5,6]	-	[21,23,17,14,15]
	3	[1,3,4,5,2,3]	-	[19,20,17,24,16]
	4	[1,1,4,5,6,4]	-	[25,22,22,19,18]
	5	[6,4,2,3,1,3]	-	[30,22,18,19,18]
Offspring	1	[1,3,6,5,2,3]	[3,5,1,4,5]	[19,20,17,24,16]
	2	[6,4,2,1,1,3]	[5,6,6,6,1]	[30,22,18,19,18]
	3	[1,2,3,2,6,3]	[2,6,4,4,5]	[20,13,27,24,15]

Table 1 provides an illustrative example of offspring fitness estimation. For simplicity, we assume that there are five individuals in the population (see rows 2 to 6 of the Table 1) and there are three offspring (see rows 7 to 9 of the Table 1). Assuming that for each individual we consider three sequencing decisions and three routing decisions to form a PC, the similarity of each offspring to the five individuals in the population is shown in the fourth column in Table 1. The multi-case fitness of each offspring can be estimated based on its most similar individual, which is shown in the fifth column in Table 1. Different from other methods that use the KNN surrogate model (Nguyen et al., 2016), which is designed to estimate the standard fitness of an individual, this paper estimates the multi-case fitness.

3.6. Offspring selection strategy

The proposed offspring selection strategy is designed to achieve a fundamental balance between exploration and exploitation when populating the next generation. Unlike conventional approaches that rely solely on fitness, our mechanism explicitly considers both performance and phenotypic diversity. Fig. 4 illustrates the offspring selection procedure.

3.6.1. Filtering offspring by performance and diversity

The first stage performs local exploitation within each phenotypic niche. Offspring with identical PCs are compared, and only the offspring with the superior multi-case fitness is retained. If their multi-case fitness is equal, the offspring with the larger rule size (more building blocks) is chosen. This ensures that for every distinct behavioural pattern present in the offspring pool, we preserve the highest-quality representative. Offspring with identical PCs that fall outside these criteria are penalised and assigned an infinite fitness, effectively removing them from consideration. This stage prevents the population from being inundated with multiple copies of similar, mediocre individuals.

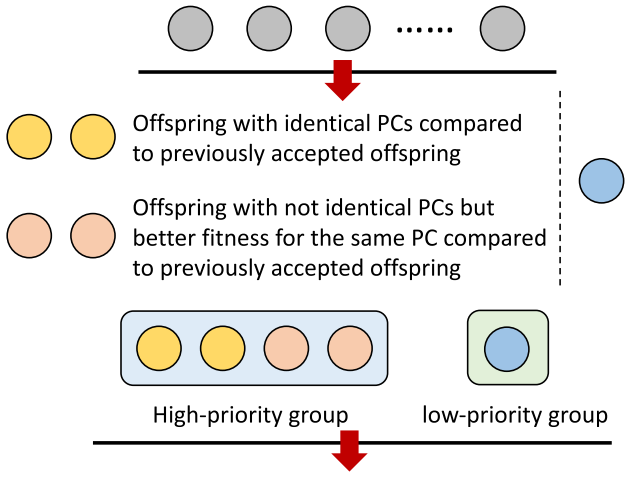

Fig. 4. The offspring selection procedure.

3.6.2. Prioritising diverse and high-performing offspring

The second stage performs global exploration by prioritising phenotypic novelty. The filtered offspring are divided into two groups based on their relationship to previously accepted offspring:

- High-priority group: Consists of (i) offspring with PCs distinct from any previously accepted offspring, and (ii) offspring sharing the same PC as a previously accepted offspring but exhibiting superior multi-case fitness (these replace the existing representative of that niche).
- Low-priority group: All remaining offspring (those with PCs already represented and inferior fitness to the retained representative).

3.6.3. Selection based on fitness and prioritisation

The final selection mechanism balances exploration and exploitation as follows. For each crossover event involving brood recombination, which generates $2 \times x$ offspring, two individuals are selected for the next generation. Selection first draws from the *high-priority group*, choosing those with the best (lowest) multi-case fitness. If the high-priority group does not contain enough candidates, the remaining slots are filled from the *low-priority group*, again prioritising the best available fitness. This procedure achieves balance through two complementary mechanisms:

- *Exploitation* occurs both in stage 1 (retaining the best individual within each phenotypic niche) and within each priority group (selecting the highest-fitness candidates when multiple are available).
- *Exploration* is driven by the preferential treatment of phenotypically novel individuals. Even if a novel individual has slightly worse fitness than some redundant candidates, it is selected first by virtue of belonging to the high-priority group. This actively introduces and preserves diverse building blocks in the population.

Comparison to conventional strategies:

- Vs. Elitism: Elitism preserves a fixed number of the globally best individuals from the current candidates based solely on fitness. Our mechanism operates on the candidate offspring pool and uses both fitness and diversity. Elitism prevents fitness loss; our mechanism shapes population composition.
- Vs. Generational replacement: Conventional generational replacement selects the top N offspring by fitness to form the next generation. This is purely fitness-driven and can rapidly lose diversity. Our mechanism may sacrifice a slightly better individual (e.g., fitness 10) to retain two distinct niches (fitnesses 12 and 14) because the long-term evolutionary potential of maintaining diverse building blocks outweighs the short-term gain of a marginally better single individual.

This offspring selection strategy is not an isolated component but is tightly integrated with DPS and brood recombination. DPS identifies complementary parents, brood recombination generates a diverse set of candidates, and our selection mechanism curates this set to preserve the most valuable combinations for future generations. This integrated approach to balancing exploration and exploitation is a key differentiator of GPDPS^o.

4. Experiment design

To verify the effectiveness of GPDPS^o, we conduct experiments across a range of DFJSS scenarios, which are defined by two key factors: (i) the objective to be optimised and (ii) utilisation level (how busy the shop

floor is). We consider three objectives described in Section 2.1 and two utilisation levels of 0.85 (less busy) and 0.95 (busier), leading to six DFJSS scenarios. Each scenario is denoted as $<\texttt{obj}, ul>$, where obj is the objective and ul denotes the utilisation level. For each scenario, we train the scheduling heuristics using GP on a set of DFJSS simulations. Then, the trained scheduling heuristics are tested on an unseen set of simulations.

4.1. Simulation model

The experiments use a simulation model (Xu et al., 2023) to investigate the factors influencing DFJSS. The simulation involves processing 5000 jobs on 10 heterogeneous machines with randomly generated processing rates between 10 and 15. Distances between machines and between the entry/exit point and each machine are sampled from a uniform discrete distribution ranging from 35 to 500, with a transport speed set at 5. New jobs arrive according to a Poisson process. The number of operations per job is randomly determined from a uniform discrete distribution between 2 and 10. Jobs are assigned weights to represent their importance: 20% have a weight of 1, 60% have a weight of 2, and 20% have a weight of 4. The work for each operation is assigned from a uniform discrete distribution ranging from 100 to 1000. Due dates for all jobs are calculated by multiplying their processing times by a factor of 1.5 and adding this time to their arrival time. The simulation begins by processing 1000 jobs to stabilise the shop floor before data collection. Performance data is then gathered from the subsequent 5000 jobs, after which the simulation concludes with a total of 6000 processed jobs.

4.1.1. Training and test instances

For each scenario, we use 50 training simulations and 50 test simulations. Each simulation is an independent execution of the DFJSS environment with a distinct random seed that determines the stochastic elements (job arrivals, processing times, etc.). During the training process, each generation utilises a different training simulation by rotating the random seed. Specifically, a new simulation instance is generated at the beginning of each generation by incrementing the seed value, ensuring that the evolving heuristics are exposed to diverse shop floor conditions throughout optimisation. This seed rotation strategy is widely adopted in large-scale DFJSS problems as it effectively balances computational efficiency with the generalisability of evolved heuristics (Zhang et al., 2021a, 2023a, 2021b). After training, the final evolved heuristics are evaluated on the 50 unseen test instances to obtain the reported test performance.

4.1.2. Relationship between simulations and algorithmic runs

For each algorithmic method, we perform 30 independent runs (as detailed in Section 4.3). In each algorithmic run, the evolved heuristic is evaluated on 50 training simulations (one per generation) and finally tested on 50 test simulations. The identical sequence of 50 training simulations and 50 test simulations is used across all 30 algorithmic runs and across all compared methods to ensure fair comparison and eliminate bias from instance variation.

This paper primarily focuses on solving the single-objective DFJSS. However, to comprehensively verify the effectiveness and robustness of the proposed method, we conducted experiments across a variety of DFJSS scenarios that consider three commonly considered scheduling objectives (i.e., max-flowtime, mean-flowtime, and max-weighted-tardiness) and shop floor settings. Evaluating the method under multiple objectives not only allows us to demonstrate its adaptability to different performance criteria but also provides stronger evidence of its generality and practical applicability in real-world manufacturing environments.

4.2. Parameter setting

In our experiments, the function set is $\{+, -, *, /, \max, \min\}$. The arithmetic operators take two arguments. The "/" operator is protected from division by zero and returns 1. The "max" and "min" functions take two arguments and return the maximum and minimum of their arguments, respectively. The terminal set of GP is shown in Table 2, which includes features related to machines (e.g., NIQ, WIQ, and MWT), operations (e.g., PT, NPT, and OWT), jobs (e.g., WKR, rDD, SLACK, NOR, W, and TIS), and transportation (e.g., TRANT). The other parameters of GP are shown in Table 3.

Table 2
Description of terminals.

Terminal	Description
TIS	Time spent in the system $= t -$ arrivalTime
W	Job weight
NOR	Remaining number of operations for the job
WKR	Remaining work
rDD	Relative due date $=$ DD $- t$
SLACK	Slack time
PT	Processing time for the operation
OWT	Waiting time for the operation $= t -$ ORT
NPT	Median processing time for the next operation
MWT	Waiting time for the machine $= t -$ MRT
NIQ	Number of operations in the machine waiting queue
WIQ	Total work in the machine waiting queue
TRANT	Transportation time

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

Table 3
The parameter settings of GPDPS^o.

Parameter	Value
Population size	1000
Number of generations	50
Total true fitness evaluations	50,000 (population × generations)
Method for initialising population	Ramped-half-and-half
Initial minimum/maximum depth	2 / 6
Elitism	10
Maximal depth	8
Crossover rate	0.80
Mutation rate	0.15
Reproduction rate	0.05
Parent selection	DPS
Terminal/non-terminal selection rate	10% / 90%
Group size g	160
Number of trials K in DPS	20
Number of brood crossovers x	5
Operator rate tuning magnitude δ	$[0.005, 0.01, 0.015, 0.02, 0.025, 0.03]$
Benefit threshold β	$[0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3]$

4.3. Comparison design

To ensure a fair comparison, all algorithms are allocated the same computational budget of 50,000 true fitness evaluations per run (50 generations × 1000 individuals evaluated per generation). True fitness evaluation, involving full simulation of 5000 jobs on 10 machines, is the dominant computational cost in DFJSS. For GPDPS^o, surrogate-based estimations (KNN with PC similarity) are computationally negligible compared to true simulation (Zhang et al., 2021b; Zhu et al., 2023), and brood recombination generates additional offspring that are evaluated using the surrogate, not true simulation. Only the two selected offspring per breeding event undergo true evaluation, maintaining the total true evaluation count at 1000 per generation. Thus, all algorithms are compared under an identical budget of true fitness evaluations, ensuring that performance differences reflect algorithmic efficacy rather than computational advantage.

Regarding the random seed control, we employ two distinct seed systems to ensure rigorous comparison. For algorithmic randomness, each independent run of every algorithm uses a different random seed for the evolutionary process (selection, crossover, mutation). For training instance randomness, all algorithms are trained on the identical sequence

of 50 training simulations across all runs. Specifically, for generation g of run r, all algorithms use the training instance generated by seed $s = 1000 \times ((g-1) \times 50 + r)$. This ensures that across all 30 runs and 50 generations, every algorithm encounters exactly the same sequence of training instances, eliminating bias from instance variation. This dual-seed strategy follows established practice in the DFJSS domain (Xu et al., 2021; Zhang et al., 2021a, 2023a).

The proposed GPDPS^o algorithm relies on two important parameters:

- δ: This parameter controls the operator rate tuning within the algorithm, indicating the magnitude of each decrease in crossover rate and corresponding increase in mutation rate when DPS fails to find a suitable donor within K attempts.
- β: This parameter defines the benefit threshold in the diverse partner selection strategy, determining the minimum level of improvement one parent must offer to the other for a pair to be considered beneficial.

Given the importance of these parameters, we perform a sensitivity analysis to determine their optimal settings for GPDPS^o. This analysis involves evaluating the algorithm's performance with different values of δ and β. Following the sensitivity analysis, we select the configuration that yields the best performance for GPDPS^o.

After identifying the optimal parameter settings for GPDPS^o, we evaluate its effectiveness by comparing it against several benchmark methods:

1. **GP** (Zhang et al., 2018a): the standard GP designed for the DFJSS problem, using tournament selection and classical crossover, with no DPS, no brood recombination, and no operator rate tuning.
2. **GPDPS^1**: A variant that uses DPS for parent selection but employs classical crossover (no brood recombination) and no operator rate tuning (i.e., without δ). This baseline isolates the effect of brood recombination when compared to GPDPS^2.
3. **GPDPS^2**: A variant that uses DPS for parent selection and the proposed brood recombination crossover (multiple offspring per parent pair), but no operator rate tuning (i.e., without δ). This baseline isolates the effect of operator rate tuning when compared to GPDPS^o.
4. **GPDPS** (Xu et al., 2022a): the original GPDPS method that employs DPS with classical crossover and does not use δ, β, brood recombination, surrogate estimation, or the proposed offspring selection strategy.
5. **GPDPS^s**: GPDPS^o with the proposed offspring selection strategy replaced by standard fitness-based selection (i.e., selecting the top two offspring by multi-case fitness without considering PC diversity), while retaining DPS, brood recombination, surrogate estimation, and operator rate tuning.
6. **GP^o**: Replaces the DPS mechanism in GPDPS^o with tournament selection (i.e., without β), while retaining brood recombination, surrogate estimation, operator rate tuning, and the proposed offspring selection strategy.
7. **GPDPS^o**: The full proposed method, incorporating DPS with benefit threshold β, brood recombination crossover, surrogate-based fitness estimation, PC-aware offspring selection, and operator rate tuning with parameter δ.

Table 4 summarizes the key components of each baseline method to clarify the ablation design:

The comparison between GP^o and GPDPS^o isolates the effectiveness of the DPS mechanism with the benefit threshold β. Comparing GPDPS^s and GPDPS^o isolates the effectiveness of the PC-aware offspring selection strategy. The comparison between GPDPS and GPDPS^o focuses on the effectiveness of the complete proposed framework relative to the closest prior work.

Comparing GPDPS^1 and GPDPS^2 isolates the effect of brood recombination, as both use DPS and lack operator rate tuning, but differ in their crossover strategy (classical vs. brood recombination). Comparing GPDPS^2 and GPDPS^o isolates the effect of operator rate tuning, as both use DPS and brood recombination, but GPDPS^o adds the δ parameter.

Table 4
Summary of baseline method components for ablation study.

Method	DPS	Brood Recombination	Operator Tuning	PC-Aware Selection	Surrogate	Classical Crossover
GP						✓
GPDPS^1	✓					✓
GPDPS^2	✓	✓				
GPDPS	✓					✓
GPDPS^s	✓	✓	✓		✓	
GP^o		✓	✓	✓	✓	
GPDPS^o	✓	✓	✓	✓	✓	

To assess and compare the performance of the algorithms, we conduct 30 independent runs for each algorithm, with each run using a different algorithmic random seed while maintaining the identical sequence of training instances across all algorithms. We employ Friedman's test (Zimmerman & Zumbo, 1993) to rank the algorithms and Wilcoxon rank-sum test (Wilcoxon et al., 1970) to identify significant performance differences between the proposed GPDPS^o and each comparison method, using a significance level of 0.05. In the subsequent results, the symbols ↑, ↓, and = denote statistically significantly better performance, significantly worse performance, and no statistically significant difference in the performance of GPDPS^o compared to the comparison method, respectively.

5. Experimental results

5.1. The effect of operator rate tuning

This section analyzes the sensitivity of the δ adjustment parameter, testing six values (0.005, 0.01, 0.015, 0.02, 0.025, 0.03) with β fixed at 0.1. For each δ, GPDPS^o is evaluated over 30 runs, and results are compared using Friedman's test. Table 5 reports the mean (standard deviation) performance across different δ values.

Based on the results of Friedman's test, there is no statistically significant difference among the proposed GPDPS^o with different δ values, and no significant trends are observed regarding how the δ parameter influences the performance of the GPDPS^o method (e.g., smaller or larger δ values do not necessarily lead to better performance). In this case, we also provide the average performance of GPDPS^o with each δ value across all six scenarios, as shown at the bottom of Table 5. It is noted that a δ of 0.02 yields the best performance among all values, with the smallest average performance of 1783.10. This is followed by δ values of 0.01 and 0.0025, with average performances of 1812.04 and 1814.55, respectively. Next are δ values of 0.03 and 0.005, with average performances of 1815.14 and 1823.89, respectively. Finally, a δ of 0.015 achieves the worst performance, with an average of 1855.46. Therefore, δ is set to 0.02 for the subsequent experiments.

Since the parameter δ dynamically adjusts the crossover rate whenever GPDPS^o struggles to find suitable parents, it directly impacts the final crossover rate at the end of each generation. To understand this effect, we analyze how the value of δ (adjustment parameter) influences the crossover rate throughout the evolutionary process. GPDPS^o considers a starting crossover rate of 0.8 for each generation. Fig. 5 presents curves illustrating the crossover rate at the end of each generation for GPDPS^o with different δ values ranging from 0.005 to 0.03 across six scenarios. From a macro perspective, a notable trend emerges: as the δ increases, the decrease in the crossover rate becomes more pronounced. Specifically, starting with an initial crossover rate of 0.8, the final crossover rate diminishes to approximately 0.7 when the δ is 0.005, and to around 0.3 when the δ is 0.03. This observed phenomenon aligns with our expectations that a larger δ makes it harder to find suitable parent pairs for crossover, leading to a more frequent decrease in the crossover rate. Another intriguing observation is that although the gap

Table 5
Mean (standard deviation) of performance of GPDPS^o with different δ values (0.005, 0.01, 0.015, 0.02, 0.025, 0.03) across six scenarios.

S	0.005	0.01	0.015	0.02	0.025	0.03
1	1276.98(12.32)	1277.90(12.03)	1277.87(15.13)	1275.26(10.38)	1280.71(14.32)	1275.38(13.08)
2	1356.70(22.12)	1348.71(10.55)	1356.23(20.09)	1352.18(17.09)	1352.96(18.15)	1351.34(19.19)
3	524.31(3.17)	524.36(3.99)	523.49(3.01)	523.99(2.60)	523.46(2.67)	524.62(2.82)
4	567.38(4.89)	566.99(3.24)	565.99(3.69)	565.93(3.74)	566.65(3.29)	566.74(2.85)
5	2168.82(58.71)	2156.06(55.42)	2182.00(44.87)	2173.59(50.81)	2181.27(47.46)	2180.21(57.07)
6	2336.23(94.04)	2305.54(82.94)	2329.89(99.64)	2308.19(75.21)	2327.90(85.78)	2287.25(77.42)
AP	1823.89	1812.04	1855.46	**1783.10**	1814.55	1815.14

* S: Scenarios; 1: <Fmax, 0.85>; 2: <Fmax, 0.95>; 3: <Fmean, 0.85>; 4: <Fmean, 0.95>; 5: <WTmax, 0.85>; 6: <WTmax, 0.95>; AP: Average performance.

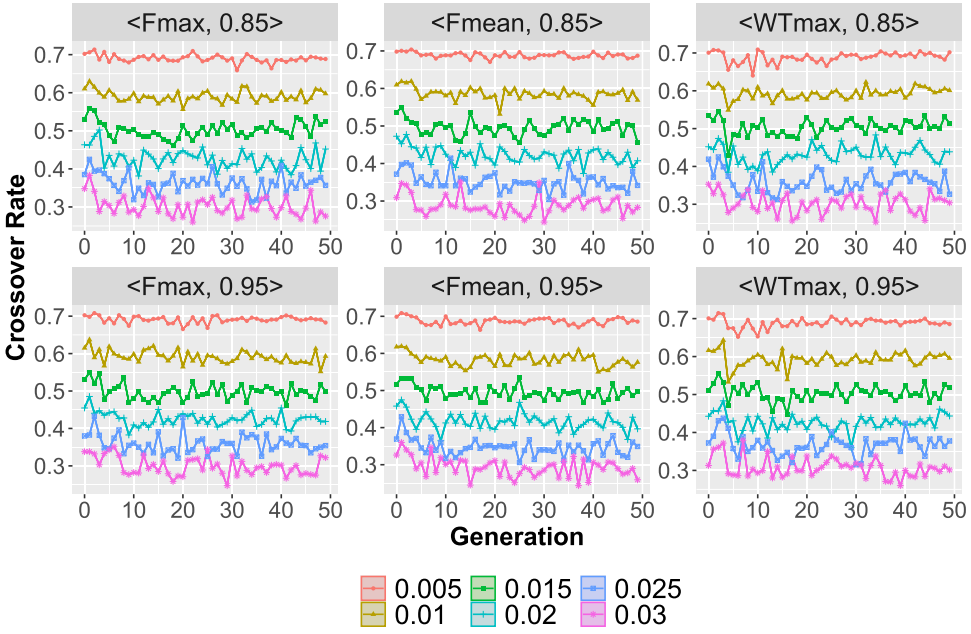

Fig. 5. Crossover rate curves at the end of each generation of GPDPS^o with different δ values (0.005, 0.01, 0.015, 0.02, 0.025, 0.03) across six scenarios. it shows that larger δ values lead to a greater reduction in crossover rate.

between the δ values remains constant (i.e., 0.005) as the δ increases, the reduction in the crossover rate becomes less substantial. This suggests that with larger δ, incremental increases have diminishing effects on the final crossover rate.

Analysing from a micro perspective, for each δ, it is evident that the crossover rate decreases progressively during the breeding process across generations. Additionally, the curves exhibit a weak converging trend: during the initial stages of the breeding process (before generation 10), the decrease in crossover rate is less pronounced compared to the middle and final stages. This phenomenon indicates that at the start of the breeding process, the adjustment of the crossover rate occurs less frequently and it is easier to identify parent pairs for crossover compared to the later stages when the algorithm approaches convergence. This pattern aligns with expectations, as the initial population is randomly generated, promoting higher diversity, whereas as the evolutionary process advances, population diversity decreases as the algorithm converges.

5.2. *The effect of excepted positive influence*

This section analyzes the sensitivity of the β adjustment parameter, testing seven values (0, 0.05, 0.1, 0.15, 0.2, 0.25 0.3) with δ fixed at 0.02. For each β, GPDPS^o is evaluated over 30 runs, and results are compared using Friedman's test. Table 6 reports the mean (standard deviation) performance for each β value.

Based on the results of Friedman's test, there is no statistically significant difference among the proposed GPDPS^o with different β values, and no significant trends are observed regarding how the β parameter influences the performance of the GPDPS^o method (e.g., smaller or larger β values do not necessarily lead to better performance). It is observed that a β value of 0.1 provides the best performance among all tested values, with the smallest average performance of 1566.43. This is followed by β values of 0.05 and 0, with average performances of 1595.06 and 1625.66, respectively. The next best are β values of 0.3 and 0.25, with average performances of 1651.26 and 1660.99, respectively. Finally, the β values of 0.2 and 0.15 show the worst performance, with averages of 1663.94 and 1682.98, respectively. Consequently, β is set to 0.1 for the subsequent experiments in this paper.

The parameter β influences the difficulty of selecting parents for crossover. A higher β value makes selecting suitable parents more challenging, which in turn affects how often the operator rate is adjusted. This ultimately impacts the final crossover rate at the end of each generation. To explore this effect, we analyse how β influences the crossover rate throughout the evolutionary process. Fig. 6 presents the crossover rate curves for GPDPS^o under β values ranging from 0 to 0.3 across six scenarios. As shown, even with δ fixed at 0.02, varying β significantly alters the final crossover rate.

From a macro perspective, a notable trend is evident: initially, during the early stages of the breeding process, all the different β values yield similar final crossover rates. However, as the evolutionary process un-

Table 6
Mean (standard deviation) of performance of GPDPS^{o} with different β values (0, 0.05, 0.1, 0.15, 0.2, 0.25 0.3) across six scenarios.

S	0	0.05	0.1	0.15	0.2	0.25	0.3
1	1275.22(13.53)	1276.53(13.32)	1275.26(10.38)	1274.76(11.60)	1282.64(14.27)	1275.99(13.92)	1275.31(10.71)
2	1348.58(17.68)	1356.31(18.19)	1352.18(17.09)	1350.49(12.67)	1357.18(20.40)	1351.16(16.42)	1351.84(16.61)
3	524.31(3.50)	524.62(3.49)	523.99(2.60)	523.99(3.12)	522.97(2.04)	524.00(2.65)	524.46(3.49)
4	566.50(3.44)	566.93(3.25)	565.93(3.74)	566.21(3.81)	566.14(4.04)	566.59(4.19)	566.41(3.12)
5	2177.39(44.19)	2170.56(46.20)	2173.59(50.81)	2175.33(58.96)	2182.31(48.87)	2160.11(60.18)	2163.15(44.33)
6	2319.64(87.76)	2311.77(91.04)	2308.19(75.21)	2357.56(110.36)	2320.98(75.65)	2312.76(95.56)	2314.16(106.53)
AP	1625.66	1595.06	**1566.43**	1682.98	1663.94	1660.99	1651.26

* S: Scenarios; 1: <Fmax, 0.85>; 2: <Fmax, 0.95>; 3: <Fmean, 0.85>; 4: <Fmean, 0.95>; 5: <WTmax, 0.85>; 6: <WTmax, 0.95>; AP: Average performance.

Table 7
The mean (standard deviation) of test performance of 30 independent runs of the proposed GPDPS^{o} and compared methods across six scenarios.

S	GP	GPDPS^{1}	GPDPS^{2}	GPDPS	GPDPS^{s}	GP^{o}	GPDPS^{o}
1	1302.82(24.70)(↑)	1296.17(16.66)(↑)	1279.05(12.08)(=)	1291.59(13.71)(↑)	1281.18(12.70)(↑)	1287.51(14.21)(↑)	1275.26(10.38)
2	1381.80(20.80)(↑)	1384.18(37.86)(↑)	1356.30(24.29)(=)	1380.29(28.22)(↑)	1359.39(18.25)(=)	1357.64(15.50)(↑)	1352.18(17.09)
3	525.14(4.04)(=)	524.91(3.90)(=)	523.36(2.80)(=)	525.08(3.68)(↑)	523.87(3.30)(=)	522.66(2.23)(↓)	523.99(2.60)
4	570.23(5.79)(↑)	568.47(4.80)(↑)	568.43(4.25)(↑)	567.62(4.22)(↑)	565.96(3.32)(=)	567.43(4.90)(=)	565.93(3.74)
5	2300.98(60.92)(↑)	2280.60(67.41)(↑)	2178.81(53.14)(=)	2275.05(71.69)(↑)	2191.82(55.88)(↑)	2184.91(61.95)(↑)	2173.59(50.81)
6	2495.21(144.71)(↑)	2493.26(148.68)(↑)	2343.95(102.85)(=)	2447.06(124.97)(↑)	2347.16(111.87)(=)	2315.83(83.44)(=)	2308.19(75.21)
↑|=|↓	5|1|0	5|1|0	1|5|0	6|0|0	2|4|0	3|0|1	-
AR	6.83	6	2.67	5	3.33	2.67	1.5

* S: Scenarios; 1: <Fmax, 0.85>; 2: <Fmax, 0.95>; 3: <Fmean, 0.85>; 4: <Fmean, 0.95>; 5: <WTmax, 0.85>; 6: <WTmax, 0.95>; AR: Average rank.

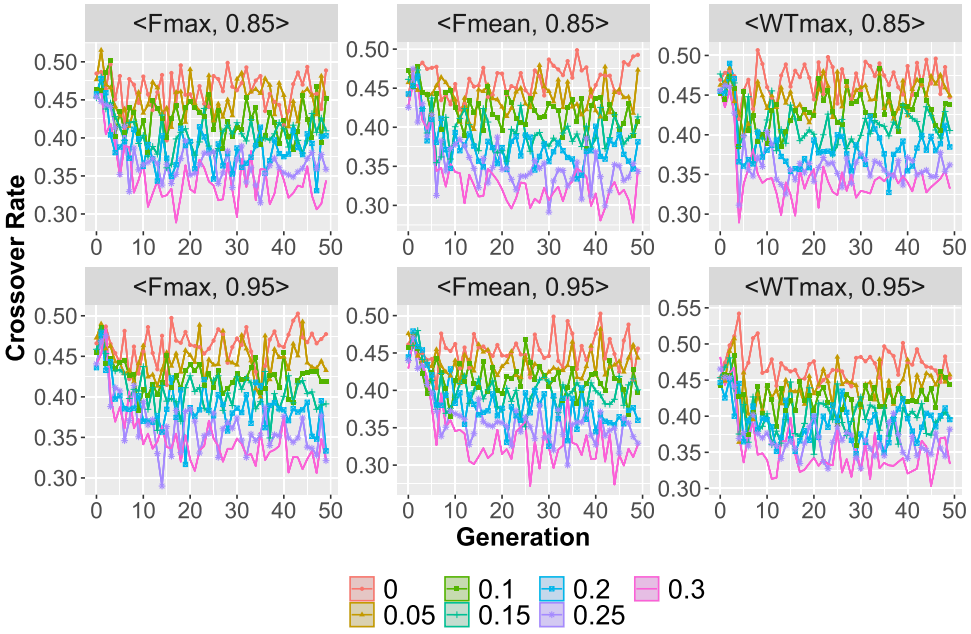

Fig. 6. Crossover rate curves at the end of each generation of GPDPS^{o} with different β values (0, 0.05, 0.1, 0.15, 0.2, 0.25 0.3) across six scenarios. It shows that higher β values result in a more pronounced decrease in crossover rate.

folds and the β increases, a more pronounced reduction in the crossover rate emerges. Specifically, starting with an initial crossover rate of 0.8, the final crossover rate diminishes to approximately 0.47 when the β is 0, and to around 0.35 when the β is 0.3. This observation suggests that with larger β, it becomes more challenging to find parent pairs where one can positively influence the other for crossover, consequently leading to a more substantial reduction in the crossover rate.

Analysing from a micro perspective, for each β, it is evident that the crossover rate decreases progressively during the breeding process across generations. Additionally, the curves exhibit a converging trend: during the initial stages of the breeding process (before generation 10), the decrease in crossover rate is less pronounced compared to the middle and final stages. This phenomenon indicates that at the start of the breeding process, the adjustment of the crossover rate occurs less frequently, making it easier to identify parent pairs for crossover compared to the later stages when the algorithm approaches convergence. This pattern aligns with expectations, as the initial population is randomly generated, promoting higher diversity, whereas as the evolutionary process advances, population diversity decreases as the algorithm converges. This phenomenon is similar to the direct manipulation of the δ parameter.

5.3. Test performance

Table 7 presents the mean (standard deviation) test performance over 30 runs of GPDPS^{o} and the compared methods across six scenarios. Friedman's test indicates statistically significant differences, with average ranks shown at the bottom. Crucially, the comparison methods are

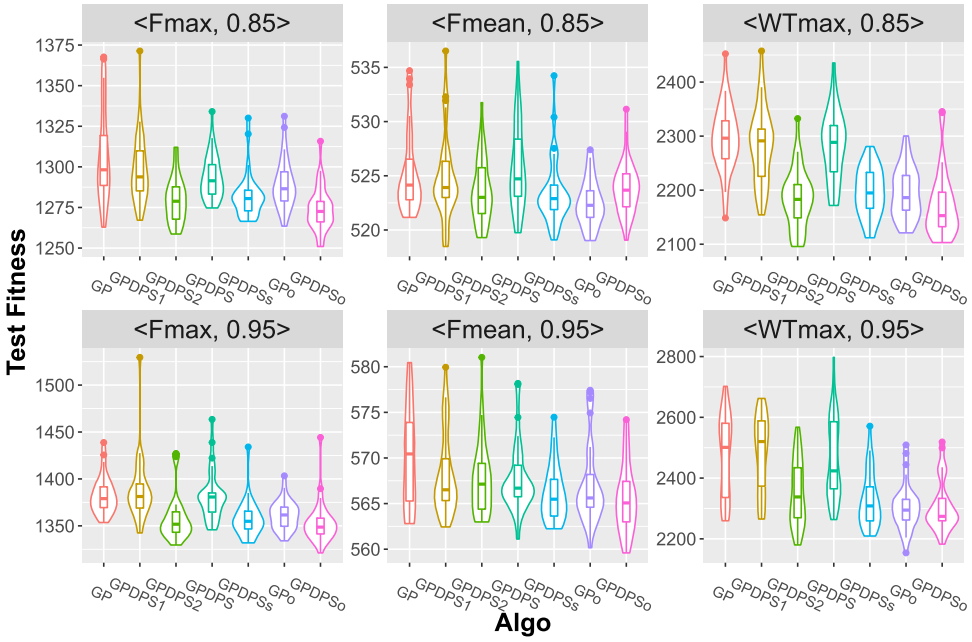

Fig. 7. Violin plots with box plots of test performance for GPDPS^o and benchmark methods across six scenarios. it shows that GPDPS^o consistently achieves superior scheduling performance.

designed as a systematic ablation study to isolate the contribution of each component of GPDPS^o. From these results, the following observations and conclusions can be drawn:

1. In comparison to all algorithms, GPDPS^o demonstrates significantly better or similar performance across all scenarios and does not exhibit significantly worse performance on any scenario. This indicates that the proposed GPDPS^o outperforms all other comparison methods on the tested DFJSS scenarios.
2. Contribution of DPS: GPDPS^o significantly outperforms GP^o on three scenarios (<Fmax, 0.85>, <Fmax, 0.95>, and <WTmax, 0.85>), shows statistically similar performance on two scenarios, and obtains significantly worse performance than GP^o on the remaining one scenario (<Fmean, 0.85>). The average rank of GPDPS^o (1.5) is substantially better than GP^o (2.67), confirming that DPS provides a clear benefit over tournament selection in identifying complementary parent pairs, particularly on max-objective scenarios where complementarity is most valuable.
3. Contribution of operator rate tuning: GPDPS^o achieves significantly better performance than GPDPS^2 on one scenario (<Fmean, 0.95>) and shows statistically similar performance on five scenarios. However, for these five scenarios with similar performance, GPDPS^o achieves consistently better (lower) objective values on four of them. The average rank difference (1.5 vs. 2.67) confirms that operator rate tuning contributes positively, particularly on more challenging scenarios where maintaining an appropriate exploration-exploitation balance is critical.
4. Contribution of brood recombination: Comparing GPDPS^2 (with brood recombination) to GPDPS^1 (without brood recombination) shows that GPDPS^2 achieves significantly better performance across all six scenarios. This demonstrates that brood recombination itself provides substantial benefits by enabling exploration of multiple genetic combinations from each complementary parent pair identified by DPS.
5. Contribution of offspring selection strategy: GPDPS^o achieves better performance than GPDPS^s on two scenarios (<Fmax, 0.85> and <WTmax, 0.85>) and achieves comparable results on the other four. The average rank difference (1.5 vs. 3.33) confirms that the PC-aware offspring selection strategy contributes to performance, particularly on max-objective scenarios where preserving diverse building blocks is critical for escaping local optima.
6. Contribution of the complete framework: GPDPS^o outperforms the original GPDPS on all six scenarios significantly. This demonstrates that the integrated framework, combining DPS, surrogate-assisted brood recombination, operator rate tuning, and PC-aware selection, provides substantial improvements over the closest prior method.
7. Synergistic effects: GPDPS^o achieves the top rank with a score of 1.5, substantially better than any ablation variant (GPDPS^2 and GP^o at 2.67, GPDPS^s at 3.33). This indicates that the components are not merely additive but synergistic: DPS identifies complementary parents, brood recombination thoroughly explores their combinations, operator rate tuning maintains balance when diversity is low, and PC-aware selection preserves valuable building blocks. Together, they create a positive feedback loop where diversity enables more effective exploitation, and exploitation informs further exploration.

Fig. 7 displays violin plots with box plots illustrating the test performance of the proposed GPDPS^o and compared methods across six scenarios. Upon comparing the shapes and widths of the violin plots among these methods, it is observed that GPDPS^o exhibits a broader and lower distribution compared to the other methods across most scenarios. This indicates that GPDPS^o consistently performs well across different test DFJSS scenarios with less variability (standard deviation). Such a trend further verifies the effectiveness of both the DPS and the new crossover strategy. The DPS efficiently selects good parent pairs, while the crossover strategy maximises the utilisation of these parents to generate high-quality offspring.

6. Further analysis

6.1. Number of unique parents selected

Unique parents selected denotes the number of distinct individuals chosen for crossover, mutation, and reproduction in each generation. This metric serves as a key indicator of population diversity during the breeding phase. A higher number of unique parents suggests a broader sampling of the available genetic material, which is characteristic of exploratory search behaviour. Conversely, a lower, more focused count

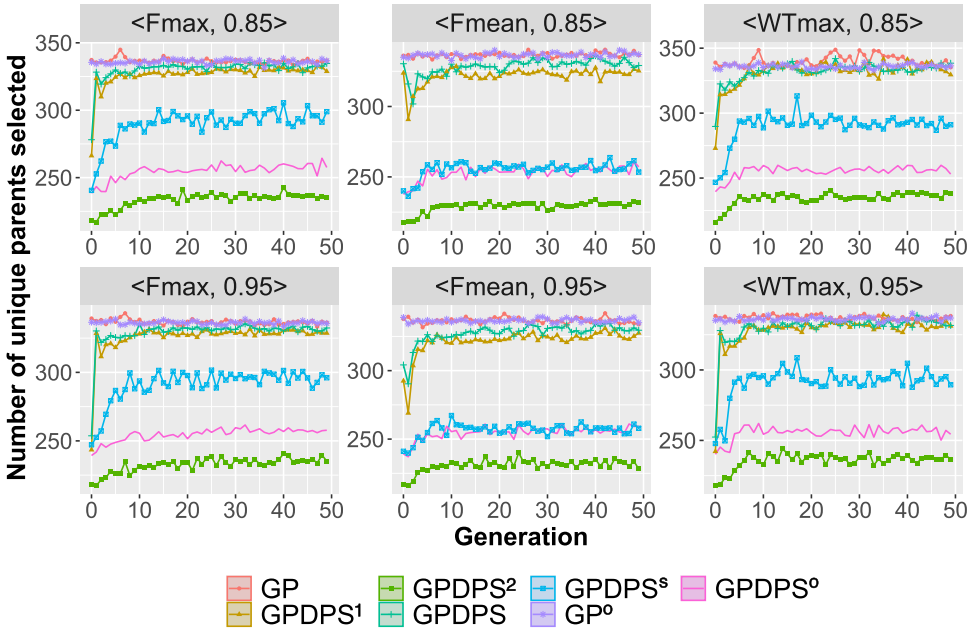

Fig. 8. The convergence curves on the *number of unique parents* selected by the proposed GPDPS^o and compared methods across six scenarios.

indicates a shift towards exploitation, where the search concentrates on refining the neighbourhood of a few promising individuals. Tracking this metric over generations therefore provides quantitative insight into the algorithm's evolving search dynamics.

Fig. 8 shows the convergence curves of unique parents selected by GPDPS^o and the compared methods across six scenarios. Observing the curves, it is evident that both GP and GP^o maintain a relatively consistent number of unique parents selected across all generations, and yield the highest count of unique parents. This consistently high count reflects their largely undirected, explorative breeding strategy. For other methods, a convergence trend is observed, where the number of unique parents starts relatively low at the beginning of the breeding process, gradually increases as generations progress, and eventually stabilises around the 10_{th} generation. However, even upon convergence, the number of unique parents remains lower than that of GP for all other methods. Specifically, both GPDPS and GPDPS^1 yield a slightly smaller number of unique parents compared to GP and GP^o. Meanwhile, GPDPS^o and GPDPS^2 yield even fewer unique parents, with GPDPS^2 producing the smallest count. In addition, it is interesting to observe that GPDPS^s exhibits different behaviours on scenarios with maximum objectives compared to mean objectives. Specifically, for maximum objectives, GPDPS^s produces a larger number of unique parents compared to GPDPS^o. However, for mean objectives, GPDPS^s results in a similar number of unique parents as GPDPS^o. This may explain why GPDPS^o performs significantly better than GPDPS^s on scenarios with maximum objectives, while showing similar performance to GPDPS^s on scenarios with mean objectives.

Interestingly, both GPDPS^o and GPDPS^2 consistently reduce the number of unique parents across all scenarios, and both use the newly developed crossover strategy, which is not employed by the other methods. This phenomenon underscores that the newly developed crossover strategy fundamentally alters the search dynamics, enabling the method to operate effectively with a more focused gene pool. By selecting fewer unique parents while maintaining or improving performance, the method demonstrates that breeding efficacy depends not merely on the quantity of parents sampled, but on how effectively their genetic material is utilised through the crossover and brood recombination mechanisms.

Furthermore, we find that the final convergence values of the number of unique parents differ for scenarios with maximum objectives and mean objectives. Generally, all methods yield a relatively larger count of unique parents on scenarios with maximum objectives compared to those with mean objectives. Upon considering test performance, it is observed that the proposed method tends to achieve superior performance more easily on scenarios with maximum objectives than on scenarios with mean objectives. This suggests that while a moderately reduced count of unique parents, as seen in GPDPS^o, can enhance performance if the selected parents are of high quality and effectively exploited, an excessive reduction, as seen in GPDPS^2, may limit genetic diversity to the point where it poses challenges in achieving better performance, particularly on more complex scenarios. This nuanced relationship between parent diversity and problem difficulty underscores the importance of a balanced approach.

6.2. Number of unique parent pairs

Number of unique parent pairs refers to the distinct pairs of parents selected for crossover in each generation. Fig. 9 shows the evolution of this number over generations for each method. As observed, GP maintains the highest count of unique parent pairs throughout the generations, indicating a diverse but potentially unfocused selection strategy that prioritises quantity over quality in parent combinations. Conversely, GPDPS exhibits the smallest number of unique parent pairs. The proposed GPDPS^o achieves fewer unique parent pairs than GP but more than GPDPS. This positions GPDPS^o in an intermediate regime, neither exhaustively sampling all possible combinations like GP, nor restricting itself as severely as GPDPS. When considering test performance, GPDPS^o, with its moderate number of unique parent pairs, consistently delivers superior performance across all scenarios. Both the proposed GPDPS^o and GPDPS exhibit a consistently and significantly smaller number of unique parent pairs compared to GP, GP^o, GPDPS^1, and GPDPS^2. Moreover, GPDPS^o has a similar number of unique parent pairs as GPDPS^s, indicating that different offspring selection strategies do not significantly affect the number of unique parent pairs. However, other strategies can influence this aspect. This analysis demonstrates that breeding efficacy depends not just on the number of parent pairs, but critically on factors such as the selection of high-quality crossover points and the generation of superior offspring via the proposed brood recombination strategy. Thus, GPDPS^o's ability to achieve superior performance with fewer parent pairs provides strong evidence that effectively leveraging a smaller

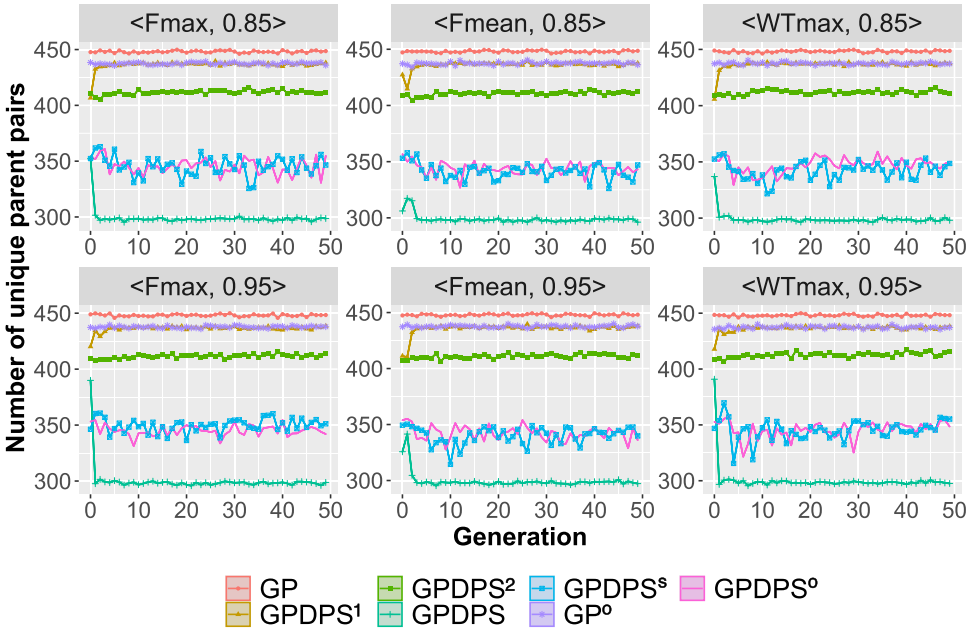

Fig. 9. The curves on the *number of unique parent pairs* of the proposed GPDPS^o and comparison methods on six scenarios.

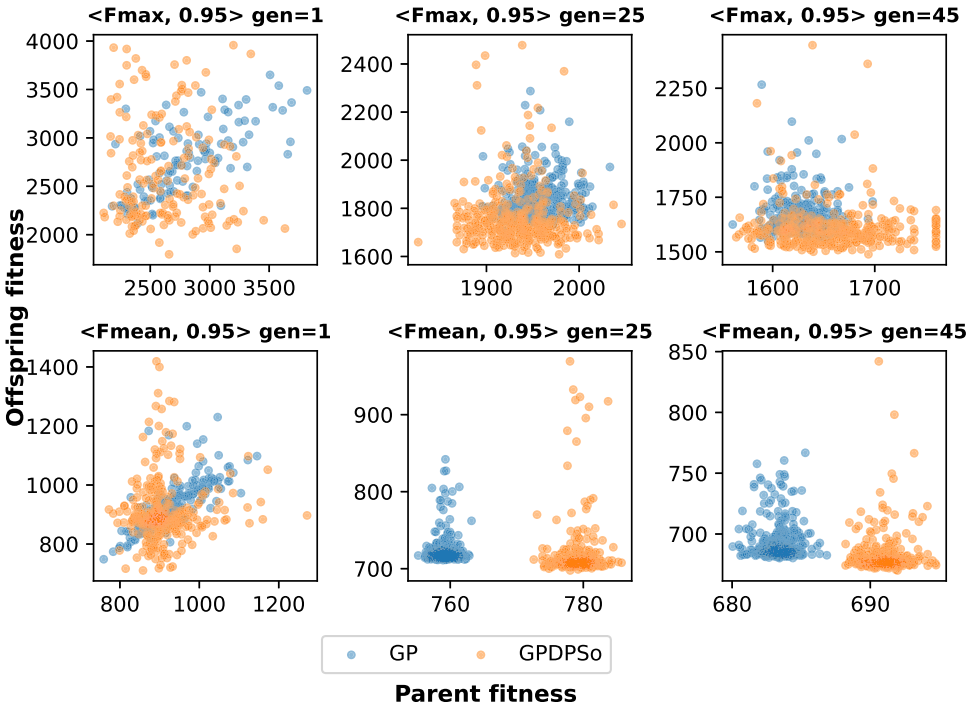

Fig. 10. The distribution of the parent versus offspring fitness of GPDPS^o and GP on two scenarios of a single run on generations 1, 25, and 45.

set of high-quality combinations can yield better results than indiscriminately exploring a larger number of combinations.

6.3. Distribution of parent fitness versus offspring fitness

To further analyse the performance of the selected parents and their offspring, we select two scenarios as examples. Fig. 10 plots the distribution of the parent fitness versus offspring fitness of GPDPS^o and GP on two scenarios of a single run on generations 1, 25, and 45. For the scenario <Fmax, 0.95>, at the beginning of the breeding process, the fitnesses of the selected parents and their offspring by the proposed methods are similar to that by the GP method. As the breeding proceeds, the fitness of the parents and the offspring selected by the proposed methods are relatively smaller than GP at the intermediate and late stages. However, for the scenario <Fmean, 0.95>, the phenomenon varies, at the beginning of the breeding process, the fitnesses of the selected parents and their offspring by the proposed methods are similar to that by the GP method. As the breeding process continues, while the fitness of the parents selected by the proposed method is bigger than GP at the intermediate and late stages, the fitness of the offspring is still relatively smaller. These findings suggest that GPDPS^o excels at selecting suitable parents to generate high-quality offspring.

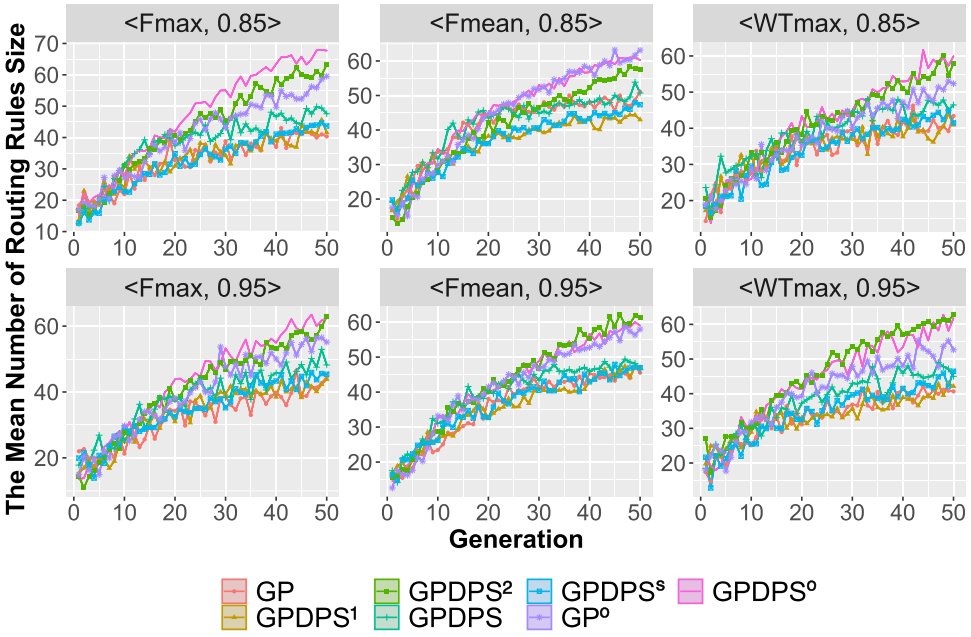

Fig. 11. The convergence curves on the *routing rule size* of the proposed GPDPS^o and comparison methods on six scenarios.

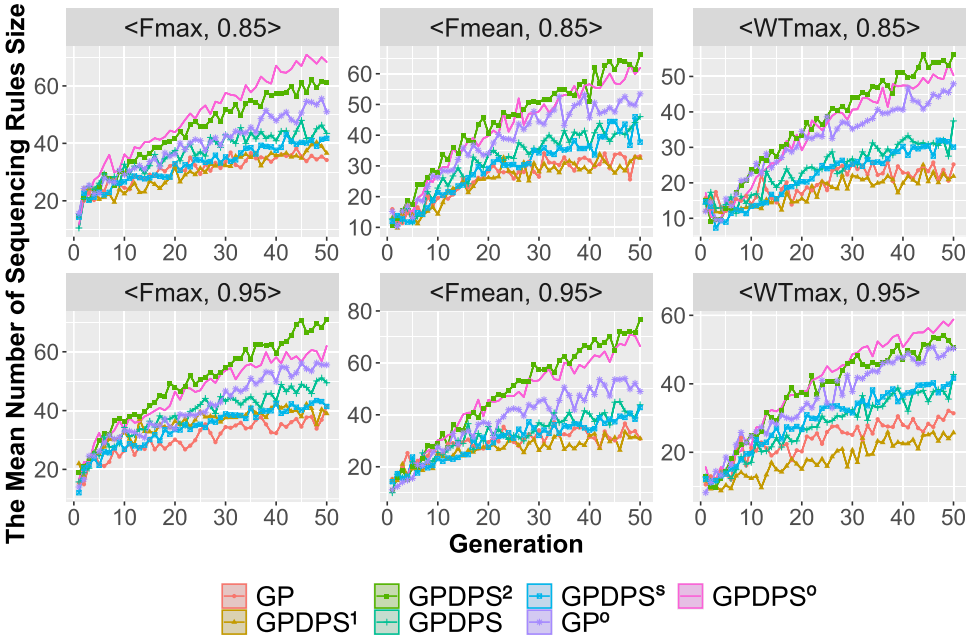

Fig. 12. The convergence curves on the *sequencing rule size* of the proposed GPDPS^o and comparison methods on six scenarios.

6.4. Rule size

This section analyses the rule size of the evolved scheduling heuristics, which serves as a measure of their complexity. Larger rules have practical significance as they enable heuristics to capture more nuanced scheduling patterns and complex decision-making relationships, thereby potentially improving performance in dynamic scheduling environments. Figs. 11 and 12 present the convergence curves of the average routing and sequencing rule size obtained from 30 independent runs of the proposed GPDPS^o and comparison methods across six scenarios. These curves illustrate how the average rule size evolves over generations for each method.

Analysing the convergence curves of rule sizes reveals an interesting trend. Initially, for all methods, the rule sizes start small (around 17 for both routing and sequencing rules) and fluctuate as the algorithms explore possible solutions. As generations progress, the rule sizes gradually increase and stabilise at higher values. Significant differences are observed in the final sizes achieved by different methods. Specifically, GPDPS^o, GP^o, and GPDPS^2 consistently generate larger sequencing and routing rules compared to other methods. This is because these three methods employ the proposed offspring selection strategy, which favors offspring with larger rules when their performance is similar, allowing them to capture more detailed scheduling patterns. It is important to note, however, that a larger rule size does not necessarily guarantee better performance. For example, in scenarios <Fmax, 0.95> and <WTmax, 0.85>, GPDPS^o achieves smaller sequencing rules than GPDPS^2 but attains better overall performance. Similarly, GPDPS^2 and GP^o show comparable results, yet GP^o produces smaller rules in most scenarios.

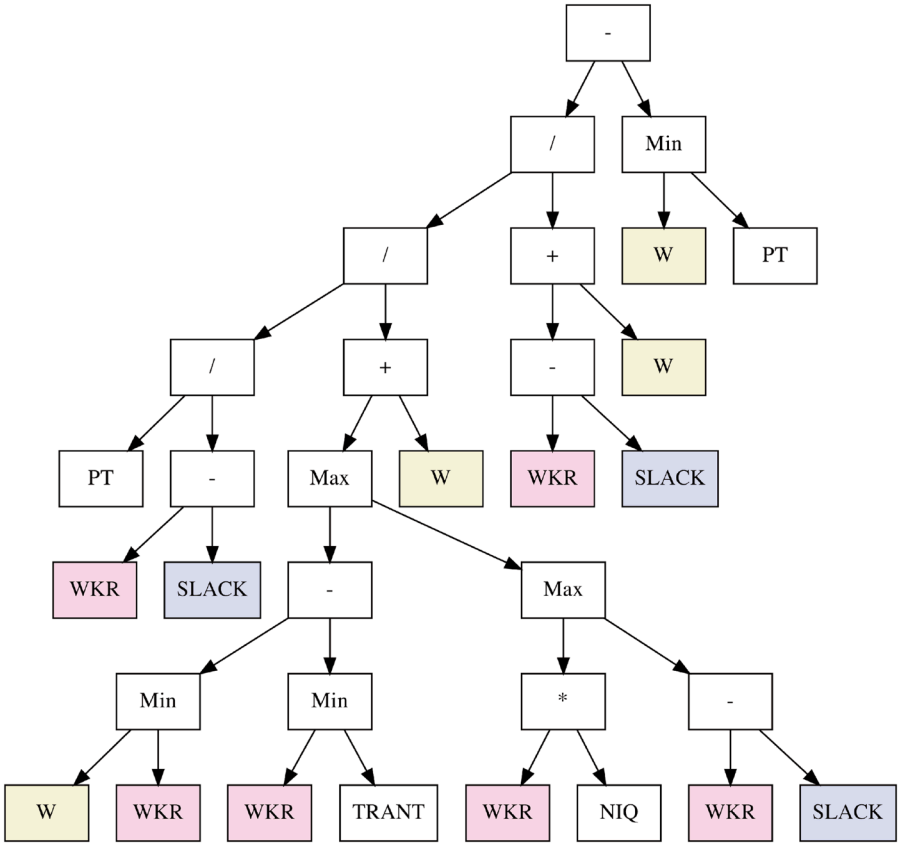

Fig. 13. An example of an evolved sequencing rule.

These observations suggest that designing heuristics with moderately large, rather than excessively large, rule sizes is sufficient to balance complexity and effectiveness. Although larger rules may reduce interpretability, this trade-off is justified as the evolved heuristics deliver strong scheduling performance and capture more sophisticated decision patterns, striking a practical balance between complexity and understandability in real-world DFJSS applications.

6.5. Structure of the evolved scheduling heuristic

To gain deeper insights into the scheduling heuristics created by GPDPS^o, we examine specific examples. Figs. 13 and 14 illustrate a sequencing rule and a routing rule, respectively, generated by GPDPS^o for the scenario <WTmax, 0.85>. This particular scheduling heuristic demonstrates promising performance on the test cases.

The sequencing rule prioritises operations based on several factors (PT, W, WKR, SLACK, NIQ, and TRANT). WKR appears most frequently (used 6 times), suggesting it has the strongest influence on the sequencing decision. W is used 4 times, indicating it's also important. SLACK appears 3 times, suggesting it has a moderate influence. PT is used twice, implying a moderate influence. NIQ and TRANT are used only once each, suggesting a minimal influence on the sequencing decision in this specific case. Since the NIQ and TRANT are the same for all operations, both can be disregarded entirely. Then, considering the max and min relationships between some factors, the sequencing rule can be simplified as S_0 in Eq. (7). This rule suggests that operations with smaller processing time (PT), smaller work remaining (WKR), larger weight (W), and larger slack are likely to be prioritised to minimise the max weighted tardiness objective.

$$S_0 = \frac{PT}{(WKR - SLACK)(WKR + W)(WKR - SLACK + W)} - W \tag{7}$$

Similar to the sequencing rule, the routing rule prioritises machines for assigning ready jobs based on several factors (TRANT, PT, W, NPT, NOR, WIQ, TIS, and WKR). TRANT appears most frequently (used 7 times), suggesting it's the most important consideration. PT is used 4 times, indicating it's also important. W is used twice, suggesting a moderate influence. On the other hand, NPT, NOR, WIQ, TIS, and WKR are used only once. Among these factors, NPT, NOT, TIS, and WKR are only related to jobs, not directly reflect machine availability, and can be ignored for machine selection in the routing process. The routing rule can be simplified as R_0 in Eq. (8), where a and b represent constants. This simplified rule suggests that machines with smaller transportation time (TRANT), smaller work remaining time in the queue (WIQ), and smaller processing time (PT) are preferred.

$$\begin{aligned} R_0 &= \frac{TRANT \times W}{NOR} - \max\{PT + WKR, TIS\} + TRANT \\ &\quad + \max\{PT, TRANT\} + WIQ + PT \times W \\ &= a \times TRANT + WIQ + b \times PT \end{aligned} \tag{8}$$

In conclusion, based on the above structure analysis of this scheduling heuristic, we conclude that four terminals (PT, WKR, SLACK, and W) play important roles in the sequencing rule, and three terminals (PT, TRANT, and WIQ) play important roles in the routing rule for minimising the max weighted tardiness objective. Moreover, it is observed that the proposed method can find specific useful terminals for the sequencing rule and the routing rule. When the sequencing decision points arrive, the information about jobs plays a decisive role. When the routing decision points arrive, the attributes of the machines, play a decisive role in the selection. Based on this visual presentation of the tree structure, we can clearly identify the importance of different terminals for different rules and allow for a more intuitive understanding of the scheduling heuristics that evolved by the proposed method.

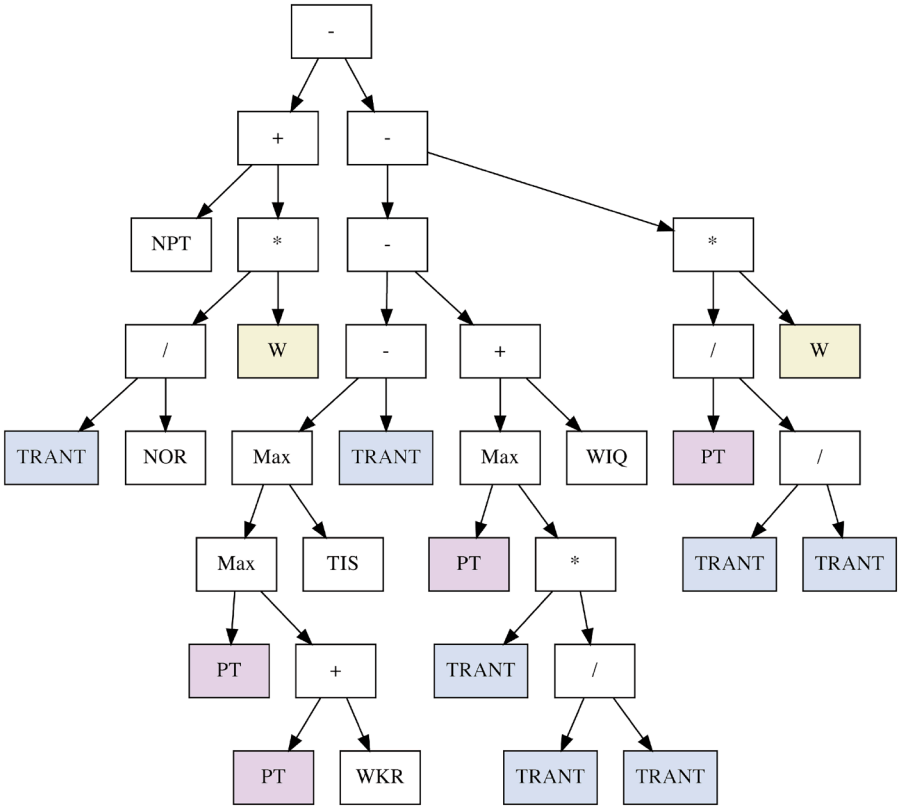

Fig. 14. An example of an evolved routing rule.

6.6. Discussion

The proposed GPDPS^o algorithm offers several clear advantages over existing approaches. Its core novelty lies in the design of a crossover mechanism that combines brood recombination, surrogate-based offspring evaluation, and an offspring selection strategy, integrated with DPS. This design enables efficient generation of high-quality offspring while preserving diversity and exploring the search space effectively. Experiments show that GPDPS^o consistently outperforms existing GP-based methods across multiple DFJSS scenarios. Further analysis reveals that GPDPS^o achieves this using fewer unique parents and pairs, indicating more efficient use of genetic material. It also evolves larger, more sophisticated scheduling heuristics, enhancing adaptability and generalization across diverse problem instances. A potential drawback is reduced interpretability due to rule complexity, though future work could address this through simplification or explainability mechanisms. Overall, GPDPS^o strikes a strong balance between solution quality, diversity, and efficiency, delivering significant improvements over existing GP approaches.

7. Conclusions

This paper presents a novel GP method, GPDPS^o, specifically designed to tackle the complexity of the DFJSS problems. GPDPS^o leverages a diverse partner selection strategy to select two individuals where one can benefit the other as parents for crossover. This strategy is complemented by crossover with brood recombination, fitness estimation, and a new offspring selection strategy to maximise the potential of the selected parents. To verify the effectiveness of GPDPS^o, this paper compared it with five GP algorithms across six diverse DFJSS scenarios defined by different objectives (max-flowtime, mean-flowtime, and max-weighted-tardiness) and utilisation levels (0.85 and 0.95). The results demonstrate that GPDPS^o consistently achieves significantly better performance or performs comparably to all compared algorithms across all six scenarios. This consistency across different performance criteria and system congestion levels provides strong evidence of the method's robustness to the most fundamental source of dynamism in DFJSS: stochastic job arrivals. The superior performance of GPDPS^o can be attributed to its ability to achieve an effective balance between exploration and exploitation, as evidenced by the detailed analysis of breeding dynamics in Sections 5 and 6. Specifically:

- Balanced use of unique parents: The analysis in Section 6.1 (Fig. 8) demonstrates that GPDPS^o maintains a moderate number of unique parents, lower than the fully explorative GP but higher than the overly restrictive GPDPS^2. This intermediate position allows GPDPS^o to focus its search on promising individuals while retaining sufficient genetic diversity to avoid premature convergence.
- Quality over quantity in parent pairs: As shown in Section 6.2 (Fig. 9), GPDPS^o operates with a moderate number of unique parent pairs. Rather than exhaustively exploring all possible combinations (as in GP) or severely restricting them (as in GPDPS), GPDPS^o focuses on generating high-quality offspring from a carefully selected set of parent pairs.
- Effective exploitation through brood recombination: The combination of focused parent selection with surrogate-assisted brood recombination enables GPDPS^o to thoroughly exploit the potential of each selected parent pair, ensuring that even with fewer breeding events, the algorithm generates and identifies high-quality offspring.
- Impact of rule size: The increased sequencing and routing rule sizes generated by GPDPS^o contribute to its superior performance, suggesting that GPDPS^o captures more complex scheduling relationships compared to other methods.

Building upon the success of GPDPS^o, several research directions can be further investigated in the future:

1. Implement self-adaptive mechanisms within GPDPS^o to automatically adjust key parameters like the expected positive influence β, or the operator rate tuning magnitude δ during the evolutionary process. This would allow the algorithm to dynamically adapt to the specific problem being solved, potentially leading to even better performance.
2. Extend GPDPS^o to handle scenarios with multiple, potentially conflicting objectives in DFJSS. This would require developing new mechanisms within the algorithm to handle trade-offs and find solutions that optimise all objectives simultaneously.
3. Investigate the robustness of GPDPS^o to additional sources of dynamism commonly encountered in real-world manufacturing, including machine breakdowns and rush orders. While the current study focuses on stochastic job arrivals, the most fundamental form of uncertainty in DFJSS, evaluating performance under these additional disruptions would further demonstrate the method's practical applicability. This would require adapting the multi-case fitness evaluation to capture performance under different disruption scenarios and potentially developing specialised surrogate models for breakdown events.

CRediT authorship contribution statement

Meng Xu: Conceptualization, Methodology, Writing – original draft; **Yi Mei:** Conceptualization, Methodology, Writing – review & editing, Supervision; **Fangfang Zhang:** Conceptualization, Methodology, Writing – review & editing, Supervision; **Yew Soon Ong:** Conceptualization, Writing – review & editing, Supervision; **Mengjie Zhang:** Conceptualization, Writing – review & editing, Supervision.

Data availability

No data was used for the research described in the article.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Meng Xu reports financial support was provided by Singapore RIE2025 Manufacturing, Trade and Connectivity (MTC) Industry Alignment Fund-Pre-Positioning. Given her role as an Associate Editor for Expert Systems with Applications (ESWA), Fangfang Zhang had no involvement in the peer review of this article and had no access to information regarding its peer review. Full responsibility for the editorial process for this article was delegated to another journal editor. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

During the preparation of this work, the author(s) used DeepSeek and ChatGPT for the purpose of language refinement. Following the use of these tools, the author(s) thoroughly reviewed and revised the content as necessary and assume(s) full responsibility for the final content of the published article.

References

Aenugu, S., & Spector, L. (2019). Lexicase selection in learning classifier systems. In *Proceedings of the genetic and evolutionary computation conference* (pp. 356–364).
Aslam, M. W., Zhu, Z., & Nandi, A. K. (2018). Diverse partner selection with brood recombination in genetic programming. *Applied Soft Computing*, *67*, 558–566.
Branke, J., Nguyen, S., Pickardt, C. W., & Zhang, M. (2015). Automated design of production scheduling heuristics: a review. *IEEE Transactions on Evolutionary Computation*, *20*(1), 110–124.
Chen, C., Ji, Z., & Wang, Y. (2018). Nsga-ii applied to dynamic flexible job shop scheduling problems with machine breakdown. *Modern Physics Letters B*, *32*(34n36), 1840111.
Day, P., & Nandi, A. K. (2008). Binary string fitness characterization and comparative partner selection in genetic programming. *IEEE Transactions on Evolutionary Computation*, *12*(6), 724–735.
De Melo, V. V., Vargas, D. V., & Banzhaf, W. (2019). Batch tournament selection for genetic programming: The quality of lexicase, the speed of tournament. In *Proceedings of the genetic and evolutionary computation conference* (pp. 994–1002).
Fang, Y., & Li, J. (2010). A review of tournament selection in genetic programming. In *International symposium on intelligence computation and applications* (pp. 181–192). Springer.
Helmuth, T., & Abdelhady, A. (2020). Benchmarking parent selection for program synthesis by genetic programming. In *Proceedings of the 2020 genetic and evolutionary computation conference companion* (pp. 237–238).
Hildebrandt, T., & Branke, J. (2015). On using surrogates with genetic programming. *Evolutionary Computation*, *23*(3), 343–367.
Jebari, K., Madiafi, M., & Elmoujahid, A. (2013). Parent selection operators for genetic algorithms. *International Journal of Engineering Research & Technology*, *2*(11), 1141–1145.
Kotanchek, M., Smits, G., & Vladislavleva, E. (2007). Pursuing the pareto paradigm: Tournaments, algorithm variations and ordinal optimization. *Genetic Programming Theory and Practice IV*, (pp. 167–185).
Koza, J. R., & Koza, J. R. (1992). Genetic programming: on the programming of computers by means of natural selection (vol. 1). MIT press.
Koza, J. R. et al. (1994). Genetic programming II (vol. 17). MIT press Cambridge, MA.
La Cava, W., Spector, L., & Danai, K. (2016). Epsilon-lexicase selection for regression. In *Proceedings of the genetic and evolutionary computation conference 2016* (pp. 741–748).
Li, R., Gong, W., Wang, L., Lu, C., & Dong, C. (2023a). Co-evolution with deep reinforcement learning for energy-aware distributed heterogeneous flexible job shop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *54*(1), 201–211.
Li, R., Gong, W., Wang, L., Lu, C., Pan, Z., & Zhuang, X. (2023b). Double DQN-based coevolution for green distributed heterogeneous hybrid flowshop scheduling with multiple priorities of jobs. *IEEE Transactions on Automation Science and Engineering*, *21*(4), 6550–6562.
Li, R., Wang, L., Gong, W., & Ming, F. (2024). An evolutionary multitasking memetic algorithm for multi-objective distributed heterogeneous welding flow shop scheduling. *IEEE Transactions on Evolutionary Computation*.
Mei, Y., Chen, Q., Lensen, A., Xue, B., & Zhang, M. (2022). Explainable artificial intelligence by genetic programming: A survey. *IEEE Transactions on Evolutionary Computation*, *27*(3), 621–641.
Metevier, B., Saini, A. K., & Spector, L. (2019). Lexicase selection beyond genetic programming. In *Genetic programming theory and practice XVI* (pp. 123–136). Springer.
Miyashita, K. (2000). Job-shop scheduling with genetic programming. In *Proceedings of the 2nd annual conference on genetic and evolutionary computation* (pp. 505–512).
Nguyen, S., Mei, Y., Xue, B., & Zhang, M. (2019a). A hybrid genetic programming algorithm for automated design of dispatching rules. *Evolutionary Computation*, *27*(3), 467–496.
Nguyen, S., Zhang, M., Johnston, M., & Tan, K. C. (2019b). Genetic programming for job shop scheduling. In *Evolutionary and swarm intelligence algorithms* (pp. 143–167). Springer.
Nguyen, S., Zhang, M., & Tan, K. C. (2016). Surrogate-assisted genetic programming with simplified models for automated design of dispatching rules. *IEEE Transactions on Cybernetics*, *47*(9), 2951–2965.
Ngwu, C., Liu, Y., & Wu, R. (2025). Reinforcement learning in dynamic job shop scheduling: a comprehensive review of AI-driven approaches in modern manufacturing. *Journal of Intelligent Manufacturing*, (pp. 1–16).
Nie, L., Gao, L., Li, P., & Li, X. (2013). A GEP-based reactive scheduling policies constructing approach for dynamic flexible job shop scheduling problem with job release dates. *Journal of Intelligent Manufacturing*, *24*(4), 763–774.
Saidi-Mehrabad, M., & Fattahi, P. (2007). Flexible job shop scheduling with tabu search algorithms. *International Journal of Advanced Manufacturing Technology*, *32*(5), 563–570.
Si, J., Li, X., Gao, L., & Li, P. (2024). An efficient and adaptive design of reinforcement learning environment to solve job shop scheduling problem with soft actor-critic algorithm. *International Journal of Production Research*, *62*(23), 8260–8275.
Spector, L., La Cava, W., Shanabrook, S., Helmuth, T., & Pantridge, E. (2018). Relaxations of lexicase parent selection. In *Genetic programming theory and practice XV* (pp. 105–120). Springer.
Stecke, K. E. (1983). Formulation and solution of nonlinear integer production planning problems for flexible manufacturing systems. *Management Science*, *29*(3), 273–288.
Tackett, W. A. (1994). Recombination, selection, and the genetic construction of computer programs. Ph.D. thesis. University of Southern California Los Angeles.
Tackett, W. A., & Carmi, A. (1994). The unique implications of brood selection for genetic programming. In *Proceedings of the IEEE congress on evolutionary computation* (pp. 160–165).
Tay, J. C., & 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.
Vilcot, G., & Billaut, J.-C. (2011). A tabu search algorithm for solving a multicriteria flexible job shop scheduling problem. *International Journal of Production Research*, *49*(23), 6963–6980.
Wilcoxon, F., Katti, S., Wilcox, R. A. et al. (1970). Critical values and probability levels for the Wilcoxon rank sum test and the Wilcoxon signed rank test. *Selected Tables in Mathematical Statistics*, *1*, 171–259.
Xie, H. (2009). An analysis of selection in genetic programming. Ph.D. thesis. Victoria University of Wellington.
Xu, M., Mei, Y., Zhang, F., & Zhang, M. (2022a). Genetic programming with diverse partner selection for dynamic flexible job shop scheduling. In *Proceedings of the genetic and evolutionary computation conference companion* (pp. 615–618).

Xu, M., Mei, Y., Zhang, F., & Zhang, M. (2023). Genetic programming with lexicase selection for large-scale dynamic flexible job shop scheduling. *IEEE Transactions on Evolutionary Computation*, *28*(5), 1235–1249.

Xu, M., Mei, Y., Zhang, F., & Zhang, M. (2024). Genetic programming and reinforcement learning on learning heuristics for dynamic scheduling: a preliminary comparison. *IEEE Computational Intelligence Magazine*, *19*(2), 18–33.

Xu, M., Mei, Y., Zhang, F., & Zhang, M. (2025). Learn to optimise for job shop scheduling: a survey with comparison between genetic programming and reinforcement learning. *Artificial Intelligence Review*, *58*(6), 1–53.

Xu, M., Zhang, F., Mei, Y., & Zhang, M. (2021). Genetic programming with archive for dynamic flexible job shop scheduling. In *Proceedings of the IEEE congress on evolutionary computation* (pp. 2117–2124). IEEE.

Xu, M., Zhang, F., Mei, Y., & Zhang, M. (2022b). Genetic programming with multi-case fitness for dynamic flexible job shop scheduling. In *Proceedings of the IEEE congress on evolutionary computation* (pp. 01–08).

Zakaria, Y., Zakaria, Y., BahaaElDin, A., & Hadhoud, M. (2021). Niching-based feature selection with multi-tree genetic programming for dynamic flexible job shop scheduling. In *Proceedings of the international joint conference on computational intelligence* (pp. 3–27).

Zhang, F., Mei, Y., Nguyen, S., Tan, K. C., & Zhang, M. (2021a). Multitask genetic programming-based generative hyperheuristics: a case study in dynamic scheduling. *IEEE Transactions on Cybernetics*, *52*(10), 10515–10528.

Zhang, F., Mei, Y., Nguyen, S., Tan, K. C., & 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., & Zhang, M. (2023a). Task relatedness based multitask genetic programming for dynamic flexible job shop scheduling. *IEEE Transactions on Evolutionary Computation*, *27*(6), 1705–1719.

Zhang, F., Mei, Y., Nguyen, S., & Zhang, M. (2021b). Collaborative multifidelity-based surrogate models for genetic programming in dynamic flexible job shop scheduling. *IEEE Transactions on Cybernetics*, *52*(8), 8142–8156.

Zhang, F., Mei, Y., Nguyen, S., & Zhang, M. (2021c). Correlation coefficient based recombinative guidance for genetic programming hyper-heuristics in dynamic flexible job shop scheduling. *IEEE Transactions on Evolutionary Computation*, *25*(3), 552–566.

Zhang, F., Mei, Y., Nguyen, S., & Zhang, M. (2021d). 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., & Zhang, M. (2022b). 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* (pp. 1218–1225).

Zhang, F., Mei, Y., Nguyen, S., & Zhang, M. (2024). 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., & Tan, K. C. (2021e). 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., Nguyen, S., Zhang, M., & Tan, K. C. (2021f). Surrogate-assisted evolutionary multitasking genetic programming for dynamic flexible job shop scheduling. *IEEE Transactions on Evolutionary Computation*, *25*(4), 651–665.

Zhang, F., Mei, Y., & Zhang, M. (2018a). Genetic programming with multi-tree representation for dynamic flexible job shop scheduling. In *Proceedings of the Australasian joint conference on artificial intelligence* (pp. 472–484). Springer.

Zhang, F., Mei, Y., & Zhang, M. (2018b). Surrogate-assisted genetic programming for dynamic flexible job shop scheduling. In *Proceedings of the Australasian joint conference on artificial intelligence* (pp. 766–772).

Zhang, F., Nguyen, S., Mei, Y., & Zhang, M. (2021g). Genetic programming for production scheduling: an evolutionary learning approach. Singapore: Springer.

Zhang, F., Shi, G., & Mei, Y. (2023b). Interpretability-aware multi-objective genetic programming for scheduling heuristics learning in dynamic flexible job shop scheduling. In *Proceedings of the IEEE congress on evolutionary computation* (pp. 1–8).

Zhang, X., & Zhu, G.-Y. (2025). A literature review of reinforcement learning methods applied to job-shop scheduling problems. *Computers & Operations Research*, *175*, 106929.

Zhang, Z., Wu, F., Qian, B., Hu, R., Wang, L., & Jin, H.-P. (2023c). A q-learning-based hyper-heuristic evolutionary algorithm for the distributed flexible job-shop scheduling problem with crane transportation. *Expert Systems with Applications*, *234*, 121050.

Zhao, F., Bao, H., Wang, L., Xu, T., Zhu, N. et al. (2022a). A heuristic and meta-heuristic based on problem-specific knowledge for distributed blocking flow-shop scheduling problem with sequence-dependent setup times. *Engineering Applications of Artificial Intelligence*, *116*, 105443.

Zhao, F., Wang, Z., & Wang, L. (2022b). A reinforcement learning driven artificial bee colony algorithm for distributed heterogeneous no-wait flowshop scheduling problem with sequence-dependent setup times. *IEEE Transactions on Automation Science and Engineering*, *20*(4), 2305–2320.

Zhou, Y., Yang, J., & 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., & 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* (pp. 384–392).

Zimmerman, D. W., & Zumbo, B. D. (1993). Relative power of the wilcoxon test, the friedman test, and repeated-measures ANOVA on ranks. *The Journal of Experimental Education*, *62*(1), 75–86.