Pareto Set Learning through Genetic Programming for Multi-Objective Dynamic Scheduling

Meng Xu, Yi Mei, *Senior Member, IEEE,* Fangfang Zhang, *Member, IEEE,* Yew Soon Ong, *Fellow, IEEE,* and Mengjie Zhang, *Fellow, IEEE*

***Abstract*—The multi-objective dynamic flexible job shop scheduling (MO-DFJSS) problem is crucial in modern manufacturing, impacting productivity and operational costs. Genetic Programming (GP) has emerged as a prominent method for MO-DFJSS due to its ability to evolve real-time responsible and effective scheduling heuristics. However, existing GP approaches often learn multiple heuristics for different regions of the Pareto front, making their management and selection complicated in real-world applications. This paper proposes a novel Pareto set learning GP (PSLGP) framework that addresses this limitation by learning a single, preference-conditioned heuristic that encompasses the entire Pareto front based on user preferences. This simplifies scheduling and allows for real-time adaptation to user-defined priorities. The framework employs a novel preference-conditioned heuristic representation that incorporates user preferences as additional inputs, enabling dynamic heuristic adjustments. To efficiently evaluate fitness without increasing training time, a surrogate model is used to estimate individual performance across different preferences, and three new fitness aggregation strategies are designed to ensure effective heuristic alignment across the Pareto front. Experimental results demonstrate that PSLGP significantly outperforms the state-of-the-art multi-objective GP approach, particularly in less busy MO-DFJSS environments, providing a more adaptable and efficient solution for dynamic scheduling challenges. Further analyses of preference influence, solution distribution, and heuristic structure provide evidence that the proposed PSLGP effectively learns preference-conditioned scheduling heuristics that align user preferences with various regions of the Pareto front.**

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

I. Introduction

Multi-objective dynamic flexible job shop scheduling (MO-DFJSS) [1] is critical in modern manufacturing, where the real-time optimization of job assignments and sequencing can greatly enhance operational efficiency, productivity, and cost-effectiveness. In MO-DFJSS, scheduling decisions navigate multiple competing objectives—such as minimizing flowtime and reducing tardiness—under variable and unpredictable conditions [2]. This complexity is further exacerbated by the need to make real-time adjustments to unforeseen new jobs' arrival over time [3].

Existing approaches to job shop scheduling are typically divided into four categories: exact methods, meta-heuristics, hand-crafted heuristics, and hyper-heuristics [4]. *Exact methods*, such as branch-and-bound [5] and dynamic programming [6], aim for optimal solutions through exhaustive searches but are often computationally expensive and impractical for large or complex problems [7], especially in dynamic multi-objective contexts. *Meta-heuristics*, like genetic algorithms [8], particle swarm optimization [9], and ant colony optimization [10], are popular for finding near-optimal solutions in reasonable timeframes, but they may struggle with robustness in highly dynamic environments [4]. *Hand-crafted heuristics* provide quick, rule-based solutions, such as the dispatching rules of shortest processing time [11] and earliest due date [12]. Although these heuristics are computationally efficient and well-suited for real-time applications, they demand significant domain knowledge and time for their design, and they often lack the flexibility and adaptability needed for specific conditions or multi-objective requirements [13]. *Hyper-heuristics* operate at a higher abstraction level, focusing on selecting or generating heuristics rather than directly solving the scheduling problem [14]. Reinforcement learning [15] and genetic programming (GP) [16] are two typical hyper-heuristic approaches for DFJSS. While these methods are beneficial in dynamic environments, the scheduling heuristics learned through reinforcement learning often lack the interpretability found in those developed using GP [17]. GP has emerged as a prominent method for addressing DFJSS problems, primarily due to its capacity to evolve effective and interpretable scheduling heuristics [4].

Existing GP approaches for MO-DFJSS integrate GP with two types of popular multi-objective techniques: non-dominated sorting (e.g., non-dominated sorting genetic algorithm II (NSGA-II) [18] and strength Pareto evolutionary algorithm 2 (SPEA2) [19]) and decomposition methods (e.g., decomposition-based multi-objective evolutionary algorithm (MOEA/D) [20]). These integrations have led to the development of NSGP-II [21], SPGP2 [21], and MOGP/D [22]. Among these, NSGP-II stands out as the leading state-of-the-art method for MO-DFJSS. However, these approaches

This research is supported by the MBIE Endeavour Smart Idea grant RTVU2305, the Distributed Smart Value Chain programme which is funded under the Singapore RIE2025 Manufacturing, Trade and Connectivity (MTC) Industry Alignment Fund-Pre-Positioning (Award No: M23L4a0001), the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG3-RP-2022-031), and the College of Computing and Data Science, Nanyang Technological University. Meng Xu is with the Singapore Institute of Manufacturing Technology, Agency for Science, Technology and Research, Singapore (e-mail: xu_meng@simtech.a-star.edu.sg). Yew Soon Ong is with the College of Computing and Data Science, Nanyang Technological University, and the Centre for Frontier AI Research, Institute of High Performance Computing, Agency for Science, Technology and Research, Singapore (e-mail: asysong@ntu.edu.sg). Yi Mei, Fangfang Zhang, and Mengjie Zhang are with the Centre for Data Science and Artificial Intelligence & School of Engineering and Computer Science, Victoria University of Wellington, Wellington 6140, New Zealand (e-mail: yi.mei@ecs.vuw.ac.nz; fangfang.zhang@ecs.vuw.ac.nz; mengjie.zhang@ecs.vuw.ac.nz).

typically learn multiple heuristics that cover different regions of the Pareto front. While this multi-heuristic strategy effectively broadens the solution spectrum, it introduces practical challenges. Managing several heuristics can increase complexity, and selecting the appropriate heuristics for real-time applications can hinder adaptability, especially in dynamic environments where preferences and priorities frequently change. Instead, learning a single heuristic that incorporates preferences as additional inputs and can dynamically and rapidly respond to real-time changes in preferences would be more efficient and flexible. However, the challenge lies in how effectively integrating preferences as additional inputs into the heuristic and how to ensure that the learned heuristic can generalize well across diverse preferences.

Motivated by these limitations and challenges, this paper presents a novel Pareto set learning GP (PSLGP) framework designed to tackle MO-DFJSS problems. The primary goal of PSLGP is to simplify the scheduling process by learning a single preference-conditioned scheduling heuristic that can dynamically adapt to various user-defined preferences in real-time, thereby covering the entire Pareto front. This approach contrasts with traditional methods by enabling one heuristic to respond flexibly to changing preferences, making PSLGP easier to manage and better suited to real-world demands. Specifically, the contributions of this paper are as follows:

1) This paper proposes a novel heuristic representation, the preference-conditioned scheduling heuristic, which integrates user preferences as inputs to enable dynamic adaptation to diverse user requirements. Additionally, this paper proposes an innovative PSLGP framework designed to learn a **single preference-conditioned scheduling heuristic** for tackling the MO-DFJSS problem, incorporating real-time user preferences as additional inputs.
2) To improve efficiency and minimize computational demands, PSLGP employs a surrogate model for evaluation, allowing for the estimation of individual performance across multiple preferences without significantly increasing training time.
3) This paper proposes three novel fitness aggregation strategies for PSLGP to ensure the learned heuristic effectively aligns with the Pareto front across a range of preferences.
4) Experimental results demonstrate that PSLGP surpasses the state-of-the-art multi-objective GP approach significantly, particularly in less busy environments.
5) Further analyses of preference influence, solution distribution, and heuristic structure confirm that PSLGP can effectively learn preference-conditioned scheduling heuristics that align with the Pareto front. This research represents a new research direction in dynamic, multi-objective scheduling, offering a more flexible, user-centric, and computationally efficient solution to the MO-DFJSS problems.

II. Background

A. Dynamic Flexible Job Shop Scheduling

DFJSS [3] involves scheduling a set of jobs $\mathcal{J} = \{J_1, J_2, ..., J_n\}$ that arrive over time on a shop floor equipped with a set of machines $\mathcal{M} = \{M_1, M_2, ..., M_m\}$. Each machine M_k has a unique processing rate γ_k. Each job J_i has an arrival time r_i, a weight ρ_i, a due date d_i, and a sequence of operations $[O_{i,1}, O_{i,2}, \ldots, O_{i,p_i}]$. Each operation $O_{i,j}$ of job J_i has a workload $\pi_{i,j}$, an optional set of machines $\mathcal{M}_{i,j} \subseteq \mathcal{M}$ on which it can be processed, and its processing time $t_{i,j,k}$ on machine M_k is defined as $t_{i,j,k} = \pi_{i,j}/\gamma_k$. Moreover, as machines are located at different geographic sites, a transportation time τ_{k_1,k_2} is required to transfer a job between machines M_{k_1} and M_{k_2}. During the scheduling process, the following constraints need to be considered:

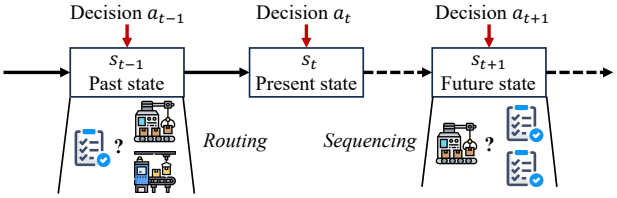

Fig. 1: The sequential decision-making process of a flexible JSS.

- Operation precedence: Operations must follow a predefined sequence; operation $O_{i,j}$ can only start after the completion of $O_{i,j-1}$ (if $j > 1$).
- Machine assignment: Each operation $O_{i,j}$ must be assigned to exactly one machine from its set of optional machines $\mathcal{M}_{i,j}$.
- Machine capacity: A machine M_k can only process one operation at any given time.
- Non-preemption: Once an operation starts on a machine, it must complete before the machine can switch to another operation.

DFJSS is a sequential decision-making problem [23] wherein scheduling decisions must be made at multiple decision points throughout the process to minimize the objectives. [24]. Fig. 1 illustrates the sequential decision-making process. DFJSS has two types of decision points: routing and sequencing. As depicted in Fig. 1, at a routing decision point, a decision is needed regarding which machine to select for processing the ready operation. At a sequencing decision point, a decision is required regarding which operation to choose as the next task for an idle machine to process. At each decision point t, the scheduling state is determined, with the present state s_t becoming active following the decision a_{t-1} made in the past state s_{t-1}. Then, based on the present state s_t, a decision a_t is necessary to advance the scheduling process to a future state s_{t+1} until the entire scheduling process is completed. The goal of MO-DFJSS is to optimize certain objectives, such as minimizing flow time and tardiness, while adhering to constraints related to machine capacity, operation precedence, and machine availability. This paper focuses on four common objectives: minimizing max-flowtime ($Fmax$), max-tardiness ($Tmax$), max-weighted-flowtime ($WFmax$), and max-weighted-tardiness ($WTmax$). Their mathematical formulations are presented below.

- $Fmax = \max_{i=1}^{n}(c_i - r_i)$
- $Tmax = \max_{i=1}^{n}\max(0, c_i - d_i)$
- $WFmax = \max_{i=1}^{n}(\rho_i \cdot (c_i - r_i))$
- $WTmax = \max_{i=1}^{n}(\rho_i \cdot \max(0, c_i - d_i))$

where c_i represents the completion time of the job J_i.

For MO-DFJSS, this paper investigates different combinations of the above objectives to create various MO-DFJSS scenarios, assessing the performance of the proposed method across these diverse scenarios.

B. Pareto Set Learning

The concept of Pareto set learning is introduced by Lin et al. in 2022 [25], [26], who apply it to multi-objective combinatorial optimization problems like traveling salesman problems, vehicle routing problems, and knapsack problems. Their approach employs reinforcement learning to construct a Pareto set by integrating multiple decision-maker preferences in real-time. The framework features complex architectures that include multiple attention layers, hundreds of dimensions in hidden layers, and various neural network components. They employ fully connected graph neural network, which is effective for small-scale and static problems but entails complex architectural design choices. It requires extensive domain knowledge, frequent updates to the graph neural networks, and specific structural configurations during the learning process, which can restrict its flexibility in handling diverse problem landscapes, particularly in large-scale and dynamic contexts such as MO-DFJSS. Additionally, the reliance on neural network methods raises concerns about interpretability [17]; the complex nature of the model makes it difficult for practitioners to understand the decision-making process behind the generated Pareto set. This lack of interpretability can limit the practical application of the method in scenarios where insights into the decision-making process are crucial [17].

In contrast, GP-based approaches provide superior adaptability and interpretability [17]. The capability of GP to simultaneously evolve both the structure and parameters of solutions in a tree-based format enables flexible adjustments across various scheduling contexts without being constrained by predefined architectures [4]. This flexibility reduces the reliance on extensive domain-specific structural design knowledge while enhancing interpretability. Moreover, the tree-based structure not only improves interpretability but also facilitates the seamless embedding of preference information as part of the inputs, enabling flexible and context-aware decision-making. Additionally, GP inherently generates a population of heuristics, diversifying the search process, mitigating local optima, and enhancing adaptability in DFJSS environments where preferences of objectives frequently shift. This adaptability, combined with the capability to evolve heuristics that directly encode preference information, distinguishes GP as a uniquely effective approach for such problems. Furthermore, GP is a dominant method in the DFJSS domain and its superiority over reinforcement learning in this area has been validated [27]. These attributes make GP an ideal method for Pareto set learning in MO-DFJSS, enabling efficient and interpretable heuristics tailored to real-time user preferences. Consequently, this paper explores the use of GP for Pareto set learning in the context of MO-DFJSS.

C. Related Work

Existing multi-objective optimization methods for DFJSS have primarily focused on learning a set of heuristics to identify a finite collection of Pareto solutions for approximating the Pareto set [21], [22], [28]–[33]. These approaches typically incorporate two popular multi-objective techniques: Pareto-dominance-based and decomposition-based methods.

Specifically, in [21], [28], [29], GP is combined with two well-known Pareto dominance-based multi-objective optimization algorithms, NSGAII [34] and SPEA2 [35], to form NSGP-II and SPGP2. These frameworks are designed to evolve scheduling heuristics for addressing the MO-DFJSS problem. Experimental results indicate that NSGP-II outperforms SPGP2 in terms of both hypervolume (HV) [36] and inverted generational distance (IGD) [37] values. Beyond Pareto dominance-based methods, [22] introduced a multi-objective GP method based on decomposition (MOGP/D), which merges the advantages of MOEA/D [20] with GP to learn scheduling heuristics for MO-DFJSS. Although MOGP/D exhibits good diversity and consistency in the training and testing of the learned Pareto front of scheduling heuristics, it achieves slightly worse HV and IGD performance compared to NSGP-II. Subsequent studies have expanded on NSGP-II. For instance, [38] investigated the impact of terminal settings on NSGP-II in the context of solving MO-DFJSS. In [39], a surrogate technique was integrated into GP, showing that surrogate-assisted methods can speed up the evolutionary process and improve the quality of the learned scheduling heuristics. A novel NSGP-II approach that merges surrogate techniques with brood recombination was introduced in [40], resulting in high-quality scheduling heuristics compared to the original NSGP-II within the same training time. Furthermore, semantic techniques are applied to NSGP-II for MO-DFJSS, demonstrating effective scheduling performance [33]. Additional research has also focused on interpretability [41] and multitasking aspects [42] within MO-DFJSS. While these methods are effective in generating diverse solutions, they exhibit limitations. The reliance on multiple heuristics can complicate the decision-making process for users, as selecting the most appropriate heuristic for a specific scenario requires significant expertise and understanding of the problem landscape. This complexity can hinder real-time adaptability, particularly in dynamic environments where job priorities and conditions frequently shift.

A few attempts have been made to integrate preference information into GP for multi-objective job shop scheduling [43]. The study in [43] proposed an interactive GP with a decision support system capable of incorporating preferences into the solution process, aiming to efficiently evolve a scheduling heuristic that aligns with the decision-maker's expectations. However, this method might not be practical, as decision-makers might lack the time to assist in the training process, or they might not be part of the same team as the developers. This limitation underscores the need for more autonomous methods that effectively incorporate user preferences without requiring active input during the training phase. In this work, we aim to address this limitation by proposing a novel multi-objective GP framework that learns a **single preference-conditioned heuristic**, capable of efficiently mapping all valid trade-off preferences to the Pareto set for MO-DFJSS in real-time.

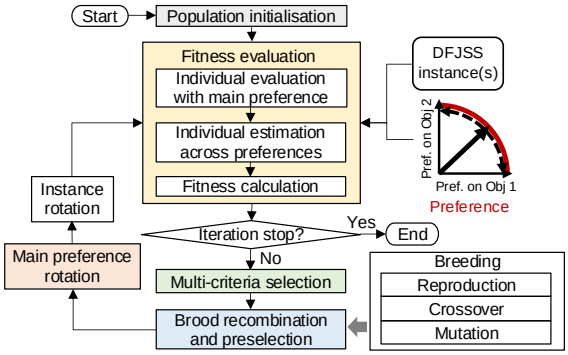

Fig. 2: The overall flowchart of the proposed method.

Algorithm 1: PSLGP Algorithm

Input: Population size N, Number of generations G, Set of preferences $\Lambda = \{\lambda_1, \lambda_2, ..., \lambda_C\}$, Set of problem instances $\Gamma = \{I_1, I_2, ..., I_G\}$
Output: Best individual (Preference-conditioned heuristic) ind_{best}

```
Initialize pop of N individuals ;                    // (Section III-B)
Initialize main preference λ_main ← λ_1; ind_best ← null;
for g ← 1 to G do
    Fitness evaluation FitnessEvaluation(pop, I_g, λ_main, Λ)
      (Algorithm 2) ;                                // (Section III-C)
    Select parents from pop using multi-criteria selection ;
      // (Section III-D)
    Generate offspring pop_new using brood recombination and
      preselection and pop ← pop_new ;  // (Section III-E)
    ind_best ← getBest(pop);
    Rotate main preference: λ_main ← λ_g ∈ Λ ;   // (Section
      III-F)
end
return ind_best
```

III. The New Method

A. Overall Framework

Fig. 2 illustrates the overall flowchart of the proposed preference-guided Pareto set learning GP (PSLGP) method and Algorithm 1 shows the pseudo-code of the PSLGP algorithm. Population initialization, fitness evaluation, selection, breeding (crossover and mutation), and instance rotation are the main processes of GP for DFJSS, whereas special designs are developed for population initialization, fitness evaluation, and selection. In addition, the brood recombination and preselection, as well as the main preference rotation are newly proposed. Overall, there are six differences between the proposed PSLGP and classical GP methods:

1) *Population initialization*: PSLGP initializes and maintains a population of individuals using newly developed representations that incorporate preference information (Section III-B);
2) *Fitness evaluation*: PSLGP develops a preference-based fitness evaluation method to assess the fitness of each individual across multiple preferences (Section III-C);
3) *Multi-criteria selection*: PSLGP designs a multi-criteria comparison strategy to select good individuals as parents (Section III-D);
4) *Brood recombination and preselection*: PSLGP proposes to adopt brood recombination and preselection to select potential good individuals as offspring for further evolution (Section III-E);
5) *Preference rotation*: PSLGP proposes to rotate the preference at each new generation, thus enhancing the individuals' generalization ability to diverse preference settings (Section III-F);
6) *Output*: PSLGP ultimately outputs a single best individual, representing a preference-conditioned scheduling heuristic, rather than a Pareto front of scheduling heuristics. This preference-conditioned scheduling heuristic can estimate the Pareto front or provide a single scheduling heuristic based on real-time preference information.

B. Individual Representation

This paper introduces a novel preference-conditioned scheduling heuristic, which serves as the representation of individuals within the population. An illustrative example of the preference-conditioned scheduling heuristic for MO-DFJSS is presented in Fig. 3. Each heuristic comprises two trees, representing the routing and sequencing rules, respectively. These tree structures are constructed through random sampling from terminals, functions, and preferences. A key innovation of this paper lies in the integration of preferences as integral components of the tree construction process. Unlike classical scheduling heuristics for DFJSS, where preferences are not explicitly considered, our approach incorporates preferences as dynamic elements that can be determined by decision-makers in real-time. This flexibility enables the preference-conditioned scheduling heuristic to adapt to varying preference settings. As depicted in Fig. 3, given a set of preferences (i.e., $\lambda_1 = (w_{1,1}, w_{2,1})$, $\lambda_2 = (w_{1,2}, w_{2,2})$, and $\lambda_3 = (w_{1,3}, w_{2,3})$) on two objectives, the preference-conditioned scheduling heuristic has the capability to generate multiple scheduling heuristics (i.e., $h_1(\cdot)$, $h_2(\cdot)$, and $h_3(\cdot)$), each potentially representing a point on the Pareto front (i.e., P_1, P_2, and P_3). Thus, rather than aiming to produce a single schedule, the preference-conditioned scheduling heuristic is designed to approximate the entire Pareto front. Furthermore, it can achieve this approximation in real time by directly incorporating preference information.

C. Fitness Evaluation

To effectively evaluate preference-conditioned scheduling heuristics across a range of preferences while aiming to approximate the Pareto front, fitness evaluation involves four main stages: 1) Individual evaluation with the main preference; 2) Non-dominated ranking; 3) Individual estimation across preferences; and 4) Fitness calculation. It should be noted that the main preference here refers to individuals with this preference undergoing accurate evaluation on the DFJSS training instance(s), while evaluations for other preferences are estimated using a surrogate model. Algorithm 2 provides the overall pseudo-code.

In the initial stage, each individual's performance is evaluated on the MO-DFJSS instance(s) using the main preference λ_{main}, which determines the primary focus of the optimization direction at each generation. In the second stage, a non-dominated ranking is conducted to assign a rank $rank_i$ to each individual ind_i based on their performance under the main preference. To reduce the excessive time costs of evaluating individuals' performance across all other preferences, the third

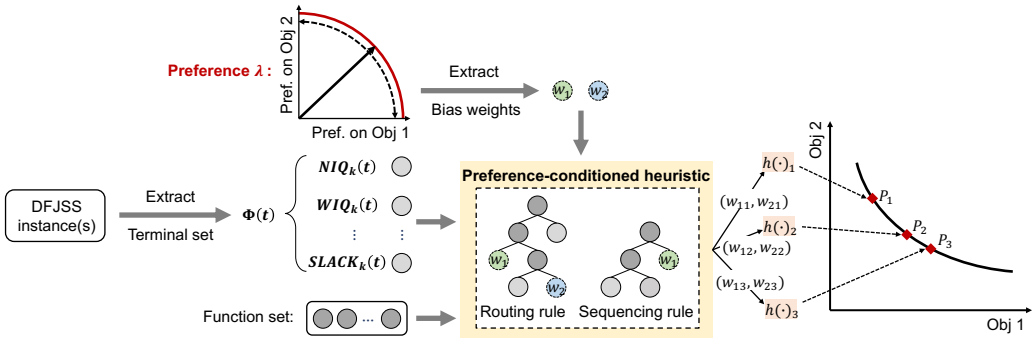

Fig. 3: An example of the preference-conditioned scheduling heuristic for MO-DFJSS. The heuristic takes a DFJSS instance as its input. The decision-makers assign their preferences (which can be seen as another type of input) on different objectives to the heuristic. The heuristic directly generates approximate Pareto solutions with different trade-offs via fast-forward inference. The generated solutions P_1, P_2, and P_3 are different optimal trade-offs between the two objectives. The ideal preference-conditioned scheduling heuristic is expected to generate solutions for all possible optimal trade-offs on the Pareto front and not generate a poor solution.

Algorithm 2: Fitness Evaluation

Input: Population pop, Set of preferences Λ, Main preference $\lambda_{main} \in \Lambda$, Problem instance I
Output: Ranks $\{rank_i\}_{i=1}^{|pop|}$, Fitness values $\{fit_i\}_{i=1}^{|pop|}$, Preference diversity values $\{predivs_i\}_{i=1}^{|pop|}$, Weighted sum objective values $\{wsums_i\}_{i=1}^{|pop|}$

Function `FitnessEvaluation`(pop, I, λ_{main}, Λ):
  for $i \leftarrow 1$ **to** $|pop|$ **do**
    $P_{i,main} \leftarrow$ `CalculateObjectives`$(ind_i, I, \lambda_{main})$;
  end
  $\{rank_i\}_{i=1}^{|pop|} \leftarrow$ `NonDominatedRank`$(\{P_{i,main}\}_{i=1}^{|pop|})$;
  for $i \leftarrow 1$ **to** $|pop|$ **do**
    $P_i \leftarrow \{P_{i,main}\}$;
    for $\lambda_j \in \Lambda \setminus \{\lambda_{main}\}$ **do**
      $estP_{i,j} \leftarrow$ `KNNModel`(ind_i, I, λ_j) (Algorithm 3);
      $P_i \leftarrow P_i \cup \{estP_{i,j}\}$;
    end
  end
  for $i \leftarrow 1$ **to** $|pop|$ **do**
    $fit_i \leftarrow \begin{cases} \texttt{CalculateHV}(P_i) \\ or\ \texttt{CalculateIGD}(P_i) \\ or\ \texttt{CalculateGD}(P_i) \end{cases}$;
    $predivi_i \leftarrow$ `PreDiversity`(Λ, P_i) (Algorithm 4);
    $wsum_i \leftarrow$ `WeightedSum`(Λ, P_i);
  end
return $\{rank_i\}_{i=1}^{|pop|}$, $\{fit_i\}_{i=1}^{|pop|}$, $\{predivi_i\}_{i=1}^{|pop|}$, $\{wsum_i\}_{i=1}^{|pop|}$

Algorithm 3: KNNModel(ind_i, I, λ_j)

Input: Individual ind_i, Problem instance I, Target preference λ_j, Population tuples with main preference (individual, objective vector, main preference) $pop_{tuple} = \{(ind_1, P_{1,main}, \lambda_{main}), ..., (ind_N, P_{N,main}, \lambda_{main})\}$
Output: Estimated objective vector $estObjs_{i,j}$ for individual ind_i with preference λ_j

Function `KNNModel`(ind_i, I, λ_j):
  $\text{PC}_i(\lambda_j) \leftarrow$ `CalculatePC`(ind_i, I, λ_j);
  $\Upsilon \leftarrow \emptyset$;
  for $(ind_k, P_{k,main}, \lambda_{main}) \in pop_{tuple}$ **do**
    $\text{PC}_k(\lambda_{main}) \leftarrow$ `CalculatePC`$(ind_k, I, \lambda_{main})$;
    $d_{i,k}(\lambda_j) \leftarrow \|\text{PC}_i(\lambda_j) - \text{PC}_k(\lambda_{main})\|$;
    $\Upsilon \leftarrow \Upsilon \cup d_{i,k}(\lambda_j)$;
  end
  $k' \leftarrow arg\min_{k \in [1,N]} \Upsilon$;
  $estP_{i,j} \leftarrow P_{k',main}$;
return $estP_{i,j}$

stage utilizes a K-nearest neighbor (KNN) surrogate model [39] to estimate performance under other preferences. KNN surrogate models are commonly used in DFJSS as efficient fitness estimators. The process involves: first, for each individual ind_i, we calculate a set of phenotypic characterizations (PC) PC_i that describe the individual's behavior across various preferences. Each PC represents a vector of values, where each value signifies the rank assigned to the candidate machine or operation by the reference rule, indicating the highest priority at a given decision point by the individual ind_i. This paper utilizes the sequencing rule and the routing rule from the best individual evolved at each generation as the reference sequencing and routing rules, respectively. To process a DFJSS instance, which typically involves thousands of decision points, we adopt a computational efficiency strategy by focusing on 20 sequencing decision points and an equivalent number of routing decision points [40]. Consequently, the PC of a scheduling heuristic comprises 40 values. Specifically, an unseen instance I is used to generate all decisions involving 7 candidate machines/operations. The decisions are then shuffled, and 20 sequencing decision points along with 20 routing decision points are randomly selected. The PC is effective in representing the phenotype/behavior of individuals within the DFJSS domain [44], serving as a reliable measure for quantifying behavioral differences among individuals.

Next, we determine the most similar individual ind_k in the population using the Euclidean distance between each PC under different preferences, defined as:

$$d_{i,k}(\lambda_j) = \|\text{PC}_i(\lambda_j) - \text{PC}_k(\lambda_{main})\| \quad (1)$$

where λ_{main} denotes the main preference. A smaller distance $d_{i,k}(\lambda)$ indicates higher similarity. The true performance of individual ind_k under the main preference is then used as an estimate for individual ind_i under the specific preference λ_j. This KNN surrogate model, therefore, provides an efficient approximation of each individual's performance across multiple preferences without exhaustive evaluations, which is shown in Algorithm 3. The final stage aggregates each individual's estimated performance across multiple preferences into a single fitness value fit_i. This paper proposes three strategies for this aggregation, based on widely-used multi-objective indicators: HV [36], IGD [37], and generational distance (GD) [45]. For an individual ind_i with performance values $P_i = \{P_{i,j}\}_{j=1}^{|\Lambda|}$ across preferences Λ ($\lambda_{main} \in \Lambda$), the three aggregation methods are shown as follows.

1) HV-based aggregation: Calculates the hypervolume HV_i

covered by the individual with respect to a reference point r, representing the volume dominated by the individual's objectives:

$$HV_i = \text{Volume}(\{P_{i,j}\} \mid r) \tag{2}$$

where r is given as the worst objective in the current population. This approach differs from the traditional HV calculation, as this paper calculates the HV not only for individuals on the Pareto front but also for all individuals in the population.

2) IGD-based aggregation: Uses the average distance between the individual's solutions and a reference Pareto front PF^*, defined as:

$$IGD_i = \frac{1}{|PF^*|} \sum_{x \in PF^*} \min_c \|P_{i,j} - x\| \tag{3}$$

where PF^* is given as the Pareto front obtained in the current population.

3) GD-based aggregation: Computes the mean distance between the individual's solutions and the closest point on the reference Pareto front:

$$GD_i = \frac{1}{|\Lambda|} \sum_{j=1}^{|\Lambda|} \min_{x \in PF^*} \|P_{i,j} - x\| \tag{4}$$

The resulting aggregated fitness $fit_i \in \{HV_i, IGD_i, GD_i\}$ serves to represent the overall performance of the individual ind_i across multiple preferences, providing a basis for selecting individuals that are robust across a diverse preference set.

The above aggregation strategy utilizes commonly used multi-objective evaluation metrics. However, certain performance values under specific preferences might not lie on the Pareto front and are therefore overlooked during evaluation. This limitation is particularly evident in metrics like HV and IGD, which primarily focus on individuals on the Pareto front. Such performance values, while not Pareto-optimal across the entire set, can still impact the effectiveness of the preference-conditioned scheduling heuristic. To capture these influences, we introduce an additional metric: the weighted-sum value $wsum_i$. Weighted-sum value $wsum_i$ represents an aggregated performance measure based on the weighted sum of normalized objectives across multiple preferences, calculated as:

$$wsum_i = \sum_j \sum_b w_{b,j} \cdot P^{\text{norm}}_{i,b,j} \tag{5}$$

where $w_{b,j}$ denotes the weight under the preference λ_j assigned to each objective b, and $P^{\text{norm}}_{i,b,j}$ is the normalised objective value for individual i under preference λ_j and objective b. This paper adopts the manual rule normalization method [22], which is normally used in the MO-DFJSS domain. This metric captures the trade-offs across multiple preferences and provides a holistic measure of the heuristic's performance.

In addition, using preferences as additional inputs might not always guarantee that evolved individuals can inherently possess these preferences within their tree structures. Moreover, ensuring that individuals with these preferences in their tree structures exhibit distinct behaviors in response to different preference settings can be challenging. To enhance the impact of preferences on the learned preference-conditioned scheduling heuristics, we propose the preference diversity $prediv_i$ metric. Preference diversity $prediv_i$ reflects the heuristic's adaptability by measuring the percentage of unique behaviors it exhibits across various preferences. It is defined as:

$$prediv_i = \frac{\text{number of unique behaviours across } \Lambda}{|\Lambda|} \tag{6}$$

The usage of preference diversity $prediv_i$ aims to address two key aspects: firstly, to enhance the influence of preferences on each scheduling heuristic, and secondly, to increase population diversity, thereby encouraging individuals with diverse combinations of preferences to coexist within the population.

Algorithm 4: Preference Diversity Calculation

Input: Individual ind_i, A DFJSS instance I, Preference set Λ
Output: Preference diversity $prediv_i$

```
1 Function PreDiversity(ind_i, I, Λ):
2     Φ_i ← ∅;
3     foreach λ_j ∈ Λ do
4         PC_i(λ_j) ← CalculatePC(ind_i, λ_j, I);
5         Φ_i ← Φ_i ∪ PC_i(λ_j);
6     end
7     prediv_i ← averageDistance(Φ_i);
8 return prediv_i
```

Algorithm 4 shows the pseudo-code of calculating the preference diversity of each individual. The input of this algorithm includes the population of individuals pop, a DFJSS instance I, and a set of preferences Λ. The DFJSS instance I is randomly generated and different from all the instances in the training set. The preference set Λ includes C pairs of preferences. The first pair remains consistent with the preference considered in the current generation, while the remaining $k = C-1$ pairs are generated by uniformly sampling from the interval $[0, 1]$. For example, when dealing with two objectives, the preferences for the remaining k pairs are determined using $\lambda_i = (w_{1,i}, w_{2,i})$, where $w_{1,i} \leftarrow i/(k-1)$ and $w_{2,i} \leftarrow (k-1-i)/(k-1)$. The uniform generation of preferences minimizes potential bias. The rationale for employing this strategy in preference generation is twofold. Firstly, it facilitates the calculation of individuals' behavior within the current generation, which is instrumental in niche management (to be discussed later). Secondly, it ensures a uniform distribution of preferences, mitigating any potential biases.

As illustrated in Algorithm 4, the *CalculatePC* function computes the PC [46], [47] for each individual (ind_i) under each preference ($\lambda_j \in \Lambda$) based on the same DFJSS instance (I) (line 4). Subsequently, a set of preference-based PCs (Φ_i) is obtained for each individual (ind_i) in the population (line 5). Traditionally, each individual is associated with a single PC used solely for measuring differences from others. However, this paper introduces a novel strategy where each individual is linked to multiple PCs, reflecting diverse preferences. In this revised context, the PC serves a dual purpose: it quantifies dissimilarities between an individual and its peers while also measuring the individual's preference diversity. After obtaining the PCs (Φ_i), the algorithm calculates the average distance (dis_i) between each pair of PCs in Φ_i and normalizes it to derive the preference diversity ($prediv_i$) for each individual (ind_i) (line 7). Fig. 4 provides a clearer example of calculating the preference diversity for an individual. By leveraging multiple PCs, we gain deeper insights into how individual behaviors

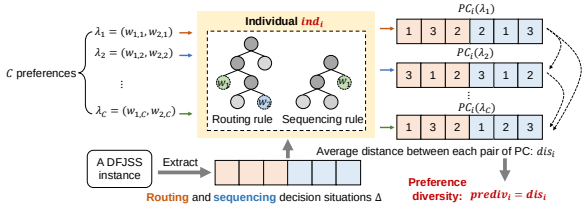

Fig. 4: An example of calculating the preference diversity.

vary in response to different preferences, thereby enriching the assessment of preference diversity within the evolutionary process.

D. Multi-Criteria Selection

The multi-criteria selection strategy selects individuals as parents through a multi-stage comparison process. This strategy applies four criteria: $rank$, $fit \in \{HV, IGD, GD\}$, $wsum$, and $prediv$. The first criterion, $rank$, is prioritized because it reflects an individual's performance under the main preference. If individuals share the same $rank$, the second criterion, $fit \in \{HV, IGD, GD\}$, is used to capture performance across multiple preferences. If fit is also equal, the third criterion, $wsum$, is applied. This considers aggregated performance across preferences and addresses potential information loss when indicators (e.g., HV and IGD) consider only non-dominated solutions, thus ignoring the influence of certain solutions. Finally, if individuals have the same $rank$, fit, and $wsum$, the fourth criterion, $prediv$, is used to select the individual with higher preference diversity, indicating a better potential for broad applicability across preferences.

E. Brood Recombination And Preselection

The proposed method introduces a novel strategy that rotates not only the instance but also the main preference for each new generation. This creates a dynamic shift in the search space, increasing the complexity of evolving effective heuristics. To adapt to this challenge, we apply brood recombination [48] to produce a large pool of candidate individuals for the next generation—specifically, generating five times the population size of candidate individuals. A preselection process then selects a subset of individuals equal to the population size from this pool, based on estimated fitness calculated with the KNN surrogate model and the individuals' PCs under the main preference, as described in Section III-C.

Specifically, candidate individuals are initially sorted based on their rank $rank$ and aggregated fitness fit. Then, the preselection process adopts a niching strategy [49], dividing individuals into distinct niches and penalizing those with similar behaviors but poorer performance within each niche. The PC ($\text{PC}_i(\lambda_{main})$) in Ψ, which uses the main preference of the current generation, is applied to measure the similarity between individuals. Each niche has a radius $\delta = 0$, so individuals with identical $\text{PC}_i(\lambda_1)$ values are grouped into the same niche. Further, the capacity κ of each niche is set to 1, meaning only one individual is maintained per niche. These settings, $\delta = 0$ and $\kappa = 1$, are typical in DFJSS for a niche configuration [47]. The final population is obtained from the sorted candidate pool, ensuring diversity preservation by focusing on top-performing, non-duplicated individuals in terms of phenotype/behavior. Importantly, the candidates' behaviors are evaluated according to the main preference of the upcoming generation, rather than the current one.

Overall, brood recombination generates a large pool of candidates, enriching the search process with a broad set of potential heuristics. The KNN surrogate model efficiently estimates fitness, enabling rapid preselection without costly evaluations. During preselection, duplicates in phenotype are eliminated, ensuring that chosen offspring are phenotypically diverse. This approach increases the likelihood of generating a wide range of solutions, a key factor in multi-objective optimization where maintaining diversity along the Pareto front is as crucial as achieving convergence. Additionally, by evaluating candidates based on the main preference of the next generation instead of the current generation, the method facilitates a smooth transition and minimizes fluctuations in the rotation of the main preference. This adaptive strategy keeps the algorithm aligned with evolving objectives and preferences, enhancing long-term performance and adaptability.

F. Main Preference Rotation

The main preference rotation strategy introduces a new main preference at each generation, enhancing the generalization of the preference-conditioned heuristic across diverse preferences. Unlike static methods that evaluate based on a single main preference throughout generations, this rotation exposes the heuristic to different main preferences over time. As a result, the heuristic becomes more adaptable, learning to perform well across a wide range of preferences. In PSLGP, the main preference plays a central role in evaluating an individual's true performance efficiently, while other preferences provide estimated evaluations for broader comparison. By rotating the main preference, we ensure that each preference receives a true performance evaluation rather than relying solely on surrogate-based estimates, which, although efficient, can not accurately capture the true performance through real evaluation. This rotation enhances the robustness of the heuristic, as it learns to optimize not only for specific instances but also to meet varying objectives across preferences. Ultimately, the main preference rotation fosters a more adaptable and widely applicable heuristic that can better handle the dynamic and diverse requirements in MO-DFJSS.

IV. Experiment Design

A. Datasets

This paper utilizes the simulation model [22] to generate datasets. This model simulates a job shop environment where 5,000 jobs are processed by 10 heterogeneous machines. Each machine's processing rate is randomly set within the range $[10, 15]$. The transportation time between machines and between each machine and the entry/exit point is sampled from a uniform discrete distribution between 7 and 100. During scheduling, jobs arrive over time according to a Poisson process. Each job consists of a randomly generated number of operations, between 2 and 10, drawn from a uniform discrete distribution. Jobs vary in importance, with 20%, 60%, and 20% of jobs assigned weights of 1, 2, and 4, respectively.

TABLE I: The scenarios.

Scenarios	Objective 1	Objective 2	Utilization
1	max-flowtime	max-weighted-tardiness	0.70
2	max-flowtime	max-weighted-tardiness	0.75
3	max-flowtime	max-weighted-tardiness	0.80
4	max-flowtime	max-weighted-tardiness	0.85
5	max-flowtime	max-weighted-tardiness	0.90
6	max-weighted-flowtime	max-tardiness	0.70
7	max-weighted-flowtime	max-tardiness	0.75
8	max-weighted-flowtime	max-tardiness	0.80
9	max-weighted-flowtime	max-tardiness	0.85
10	max-weighted-flowtime	max-tardiness	0.90

The workload for each operation is randomly assigned from a uniform discrete distribution in the range $[100, 1000]$. A due date factor of 1.5 is applied, meaning each job's due date is set at 1.5 times its total processing time from its arrival.

Utilization levels are critical for modeling different DFJSS scenarios, as higher utilization corresponds to a busier job shop environment. In this study, we examine five utilization levels: 0.70, 0.75, 0.80, 0.85, and 0.90, creating diverse MO-DFJSS scenarios across varying job shop intensities. The specific scenarios are detailed in Table I. To ensure data accuracy, the initial 1,000 jobs are used as warm-up jobs to stabilize the system and simulate realistic job shop conditions over time. Following the warm-up, data collection begins for the next 5,000 jobs, with the simulation ending after processing 6,000 jobs. To enhance the generalization of the evolved scheduling heuristics, a single replication of the simulation is used, while random seeds are varied for each GP generation, producing diverse instances within each scenario [50]. This approach helps ensure robust heuristic performance across different DFJSS scenarios.

B. Comparison Design

As this paper presents a novel research area within both GP and MO-DFJSS: the development of **a single preference-conditioned heuristic** capable of efficiently mapping all valid trade-off preferences to the Pareto set for MO-DFJSS in real-time, there is a lack of existing literature. To evaluate the proposed method, we compare it against the current state-of-the-art multi-objective approach on MO-DFJSS, NSGP-II, focusing on performance metrics including HV, IGD, and GD. To ensure a fair comparison with NSGP-II, which directly evolves a set of non-dominated heuristics, we first generate a set of 200 preferences using the Das & Dennis method [51]. This approach ensures a relatively uniform distribution of weight vectors, which is essential for obtaining a diverse and well-represented Pareto front. PSLGP is then executed for each preference, producing a corresponding heuristic for that specific preference. This resulted in a set of derived heuristics, each tailored to a different preference direction. The solutions generated by this set of derived heuristics are then used to calculate the HV, IGD, and GD metrics, enabling a direct comparison with the non-dominated set produced by NSGP-II. The methods compared are as follows:

1) NSGP-II: The leading multi-objective GP for MO-DFJSS [21].
2) PSLGP-G (Ours): The proposed preference-conditioned multi-objective GP method, primarily evaluated using GD as the performance criterion.

TABLE II: The GP terminal and function set for DFJSS.

Notation	Description
NIQ	Number of operations in the queue
WIQ	Work in the queue
MWT	Waiting time of the machine = t* - MRT*
PT	Processing time of the operation
NPT	Median processing time for the next operation
OWT	Waiting time of the operation = t - ORT*
WKR	Work remaining
NOR	Number of operations remaining
rDD	Relative due date = DD* - t
SL	Slack
W	Job weight
TIS	Time in system = t - releaseTime*
TRANT	Transportation time
Function	$+, -, \times, /, max, min$

* t: current time; MRT: machine ready time; ORT: operation ready time; DD: due date; releaseTime: release time.

TABLE III: The parameter settings for PSLGP and NSGP-II.

Parameter	Value
Population size	1000
Number of generations	50
Method for initializing 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
Terminal/non-terminal selection rate	10% / 90%
Parent selection	Tournament selection with 7

3) PSLGP-I (Ours): The proposed preference-conditioned multi-objective GP method, primarily evaluated using IGD as the performance criterion.
4) PSLGP-H (Ours): The proposed preference-conditioned multi-objective GP method, primarily evaluated using HV as the performance criterion.

C. Parameter Setting

In the experiments, the terminal set and function set used to create individuals are detailed in Table II [3]. The terminal set includes features associated with machines (e.g., NIQ, WIQ, and MWT), operations (e.g., PT, NPT, and OWT), jobs (e.g., WKR, NOR, W, TIS, rDD, and SL), and transportation (e.g., TRANT). The function set consists of arithmetic operators that operate on two arguments. The division operator "/" is protected, returning 1 when the divisor is zero. The max and min functions also take two arguments, returning the maximum and minimum values, respectively. The parameter configurations for PSLGP and the comparison method are presented in Table III [33].

V. Experimental Results

A. Test Performance

Tables IV, V, and VI present the results of four multi-objective optimization methods, NSGP-II, PSLGP-G, PSLGP-I, and PSLGP-H, across ten DFJSS scenarios, with each table focusing on a different metric: HV, IGD, and GD, respectively. Wilcoxon tests [52] are performed to statistically compare the proposed methods (PSLGP-G, PSLGP-I, PSLGP-H) against the baseline (NSGP-II), with significance marked as: $\uparrow$ for significantly better, $\downarrow$ for worse, and = for no significant

TABLE IV: The test **HV** performance of the proposed methods and the comparison method across ten DFJSS scenarios.

S[1]	NSGP-II	PSLGP-G	PSLGP-I	PSLGP-H
1	0.810(0.048)	0.792(0.069)(=)	0.834(0.067)(↑)	0.820(0.068)(=)
2	0.956(0.010)	0.947(0.024)(=)	0.958(0.012)(=)	0.959(0.015)(=)
3	0.858(0.036)	0.836(0.060)(=)	0.852(0.038)(=)	0.849(0.056)(=)
4	0.922(0.021)	0.864(0.069)(↓)	0.895(0.060)(=)	0.915(0.026)(=)
5	0.960(0.012)	0.919(0.048)(↓)	0.931(0.031)(↓)	0.936(0.023)(↓)
6	0.911(0.023)	0.927(0.018)(↑)	0.919(0.030)(=)	0.934(0.026)(↑)
7	0.942(0.013)	0.936(0.023)(=)	0.952(0.017)(↑)	0.942(0.036)(=)
8	0.920(0.015)	0.917(0.044)(=)	0.920(0.031)(=)	0.916(0.029)(=)
9	0.789(0.041)	0.762(0.105)(=)	0.815(0.076)(↑)	0.823(0.044)(↑)
10	0.942(0.014)	0.896(0.049)(↓)	0.912(0.033)(↓)	0.916(0.022)(↓)
w|d|l	-	1|6|3	3|5|2	2|6|2
r	2.6	2.8	1.8	2.8

TABLE V: The test **IGD** performance of the proposed methods and the comparison method across ten DFJSS scenarios.

S[1]	NSGP-II	PSLGP-G	PSLGP-I	PSLGP-H
1	0.159(0.033)	0.155(0.051)(=)	0.129(0.046)(↑)	0.136(0.041)(↑)
2	0.031(0.007)	0.035(0.019)(=)	0.024(0.008)(↑)	0.023(0.007)(↑)
3	0.089(0.017)	0.100(0.040)(=)	0.091(0.019)(=)	0.091(0.033)(=)
4	0.063(0.016)	0.098(0.069)(↓)	0.073(0.051)(=)	0.061(0.017)(=)
5	0.029(0.008)	0.049(0.045)(↓)	0.036(0.026)(=)	0.031(0.020)(=)
6	0.062(0.012)	0.047(0.011)(↑)	0.053(0.018)(↑)	0.044(0.016)(↑)
7	0.035(0.007)	0.036(0.015)(=)	0.027(0.009)(↑)	0.035(0.039)(↑)
8	0.057(0.007)	0.066(0.023)(=)	0.054(0.014)(=)	0.059(0.014)(=)
9	0.109(0.031)	0.126(0.077)(=)	0.086(0.048)(↑)	0.082(0.027)(↑)
10	0.031(0.009)	0.058(0.045)(↓)	0.042(0.027)(↓)	0.040(0.014)(↓)
w|d|l	-	1|6|3	5|4|1	5|4|1
r	2.7	3.0	1.8	2.5

TABLE VI: The test **GD** performance of the proposed methods and the comparison method across ten DFJSS scenarios.

S[1]	NSGP-II	PSLGP-G	PSLGP-I	PSLGP-H
1	0.169(0.054)	0.121(0.039)(↑)	0.111(0.044)(↑)	0.118(0.036)(↑)
2	0.037(0.011)	0.029(0.018)(↑)	0.019(0.009)(↑)	0.017(0.008)(↑)
3	0.091(0.032)	0.050(0.034)(↑)	0.043(0.026)(↑)	0.052(0.035)(↑)
4	0.065(0.015)	0.053(0.060)(↑)	0.041(0.046)(↑)	0.029(0.011)(↑)
5	0.042(0.015)	0.033(0.042)(↑)	0.026(0.025)(↑)	0.021(0.018)(↑)
6	0.056(0.035)	0.032(0.009)(↑)	0.037(0.017)(↑)	0.028(0.012)(↑)
7	0.041(0.028)	0.021(0.011)(↑)	0.017(0.010)(↑)	0.024(0.035)(↑)
8	0.059(0.037)	0.029(0.016)(↑)	0.033(0.010)(↑)	0.032(0.013)(↑)
9	0.085(0.028)	0.047(0.036)(↑)	0.041(0.019)(↑)	0.045(0.032)(↑)
10	0.032(0.015)	0.031(0.033)(↑)	0.022(0.018)(↑)	0.017(0.010)(↑)
w|d|l	-	10|0|0	10|0|0	10|0|0
r	3.6	2.4	1.9	2.1

difference. Additionally, Friedman tests [53] are conducted to rank the overall performance of these methods.

In terms of HV performance (as shown in Table IV), PSLGP-I consistently outperforms the other methods, achieving the highest rank (1.8), which underscores its effectiveness, particularly in scenarios that do not push utilization to extreme levels (0.90 utilization level). PSLGP-G generally matches NSGP-II's performance with a rank of 2.8 and demonstrates significantly better results in low-utilization scenarios. However, it underperforms in high-utilization scenarios, where its performance declines significantly. Similarly, PSLGP-H shares a comparable overall rank of 2.8 and follows the same trend: excelling in low-utilization scenarios but showing weaker results under higher utilization conditions. For example, in scenario <WFmax-Tmax, 0.70>, both PSLGP-G and PSLGP-H outperform NSGP-II, achieving significantly higher HV scores (↑), while PSLGP-I shows comparable results (=). In scenario <WFmax-Tmax, 0.75>, PSLGP-I again demonstrates superior HV performance (↑), confirming its effectiveness in handling WFmax-Tmax scenarios. In scenario <WFmax-Tmax, 0.85>, both PSLGP-I and PSLGP-H outperform NSGP-II (↑), indicating their ability to handle challenging conditions better than the baseline. However, in the most extreme scenario, <WFmax-Tmax, 0.90>, all proposed methods perform worse than NSGP-II (↓), highlighting NSGP-II's resilience under high workload conditions. Overall, PSLGP-I emerges as the most consistent and generally best-performing method, delivering significant improvements over NSGP-II in various scenarios. While PSLGP-G and PSLGP-H also offer competitive performance, they are less consistent and underperform in certain high-utilization cases. NSGP-II remains a strong baseline, particularly in scenarios with high utilization, where it maintains solid performance.

In terms of IGD performance (as shown in Table V), PSLGP-I consistently outperforms NSGP-II in multiple scenarios, achieving the highest rank (1.8), particularly excelling in <WFmax-Tmax> conditions, making it the best performer overall. PSLGP-H also performs well with a rank of 2.5, similar to PSLGP-I, but with a slightly lower ranking. PSLGP-G, with a rank of 3.0, generally matches NSGP-II's performance in many scenarios but falters in high-utilization settings, leading to significant underperformance. For example, in scenario <Fmax-WTmax, 0.75>, both PSLGP-I and PSLGP-H significantly outperform NSGP-II, with IGD values of 0.024 and 0.023, respectively, while PSLGP-G has slightly worse results but without a significant difference (=). In scenario <WFmax-Tmax, 0.70>, PSLGP-G, PSLGP-I, and PSLGP-H all outperform NSGP-II (↑), demonstrating significantly better IGD values. However, in the high-utilization scenario <WFmax-Tmax, 0.90>, all three proposed methods, PSLGP-G, PSLGP-I, and PSLGP-H, perform worse than NSGP-II (↓), indicating NSGP-II's robustness in extreme high-utilization conditions. Overall, PSLGP-I is the clear winner, achieving the best IGD performance in 5 scenarios, effectively reducing IGD across a wide range of conditions. PSLGP-H is a strong competitor, performing similarly to PSLGP-I in many scenarios but with a slight disadvantage in the overall ranking. PSLGP-G performs comparably in lower-utilization scenarios but struggles significantly in high-utilization environments, where it tends to underperform.

In terms of GD performance (as shown in Table VI), PSLGP-I is the strongest performer overall, with an average rank of 1.9, demonstrating consistent improvements in convergence across all scenarios. It consistently achieves lower GD values than NSGP-II, confirming its ability to approximate the Pareto front more closely. PSLGP-H also shows significant improvements in GD across all scenarios, with an average rank of 2.1, making it a strong competitor to PSLGP-I. PSLGP-G, with an average rank of 2.4, performs well across all scenarios, consistently outperforming NSGP-II in terms of GD, though it is not the top performer. Overall, PSLGP-I emerges as the best method, consistently delivering lower GD values than both NSGP-II and the other proposed methods across all scenarios. Its effectiveness in achieving close convergence

[1]S: Scenario; 1: <Fmax-WTmax, 0.70>; 2: <Fmax-WTmax, 0.75>; 3: <Fmax-WTmax, 0.80>; 4: <Fmax-WTmax, 0.85>; 5: <Fmax-WTmax, 0.90>; 6: <WFmax-Tmax, 0.70>; 7: <WFmax-Tmax, 0.75>; 8: <WFmax-Tmax, 0.80>; 9: <WFmax-Tmax, 0.85>; 10: <WFmax-Tmax, 0.90>; w|d|l: win | draw | loss; r: average rank

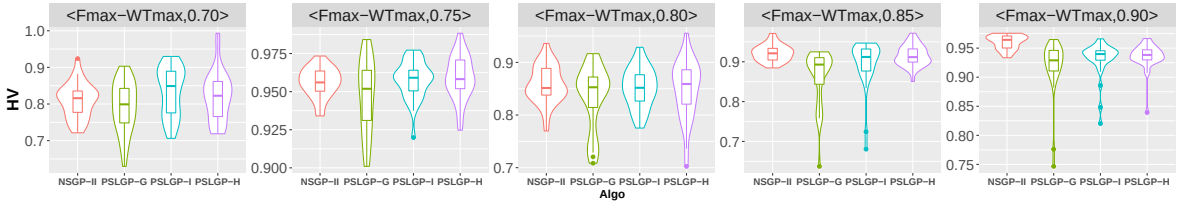

Fig. 5: The box plots of the HV performance of the proposed methods compared to the baseline methods across 5 DFJSS scenarios.

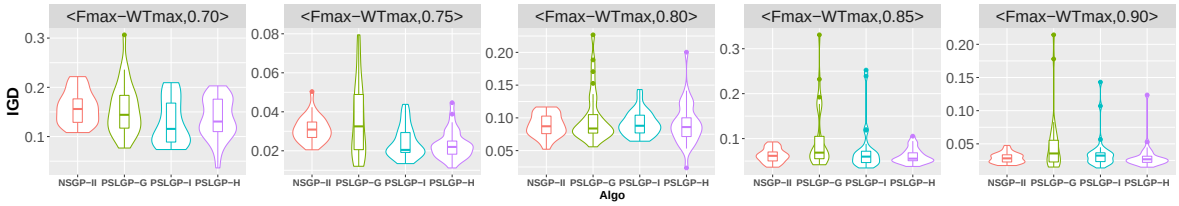

Fig. 6: The box plots of the IGD performance of the proposed methods compared to the baseline methods across 5 DFJSS scenarios.

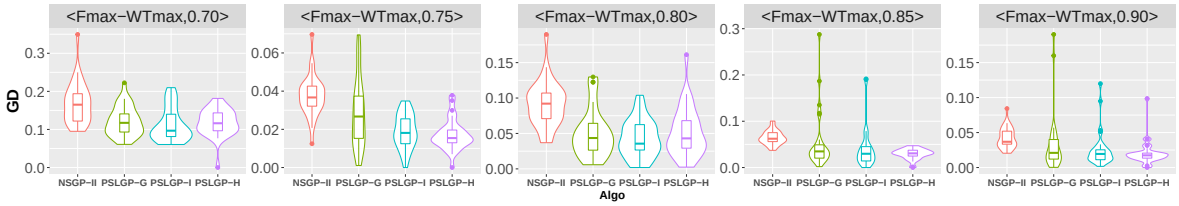

Fig. 7: The box plots of the GD performance of the proposed methods compared to the baseline methods across 5 DFJSS scenarios.

to the Pareto front at various utilization levels highlights its superior performance. All three proposed methods, PSLGP-G, PSLGP-I, and PSLGP-H, consistently outperform NSGP-II across every scenario. The unanimous wins (10-0-0) for each of the proposed methods indicate their significant advantages in reducing GD, regardless of the scenario.

While one might intuitively expect PSLGP-H to excel in HV, PSLGP-I in IGD, and PSLGP-G in GD, our empirical results show that PSLGP-I consistently achieves the best performance across all three metrics. This can be attributed to the proposed multi-criteria selection strategy and the nature of IGD. The multi-criteria selection strategy first ranks individuals based on their non-dominated rank according to the main preference. If multiple individuals share the same rank, a secondary comparison is conducted using HV, IGD, or GD. As a result, by the end of training, the best preference-conditioned individual is primarily selected based on its rank rather than a specific metric. Additionally, IGD measures the average distance from points in the Pareto front approximation to the (estimated) true Pareto front, inherently promoting both convergence and diversity. Since strong convergence and diversity benefit HV and GD as well, optimizing IGD leads to superior performance across all three metrics. In contrast, PSLGP-H, which focuses solely on HV, may generate solutions that cover a large volume but fail to converge to the true Pareto front, resulting in suboptimal IGD and GD. Similarly, PSLGP-G, which optimizes GD directly, may prioritize convergence at the expense of diversity, leading to lower HV.

To present intuitive and clearer results, we visualize the box plots of HV, IGD, and GD from 30 runs of the four methods across 5 out of these 10 scenarios, as shown in Figs. 5, 6, and 7, respectively. The results align with the previous analysis. The box plots reveal the variability of HV, IGD, and GD values within each method, where PSLGP-G, PSLGP-I, and PSLGP-H, generally exhibit taller distributions, indicating greater variability in their solutions. In contrast, NSGP-II shows shorter distributions, suggesting more consistent performance. This may be because NSGP-II learns different heuristics to cover the Pareto front, while the proposed methods focus on a single heuristic and further consider multiple preferences to achieve coverage of the Pareto front.

In summary, the comparative analysis of the four multi-objective optimization methods, NSGP-II, PSLGP-G, PSLGP-I, and PSLGP-H, across ten DFJSS scenarios reveals distinct performance trends regarding HV, IGD, and GD metrics. PSLGP-I stands out as the overall top performer, consistently attaining the highest ranks and demonstrating superior effectiveness across HV, IGD, and GD metrics, especially under low-utilization conditions. While PSLGP-H performs competitively, especially in low-utilization scenarios, its overall ranking slightly lags behind that of PSLGP-I. PSLGP-G shows strong performance in many instances but struggles significantly in high-utilization conditions, resulting in underperformance compared to NSGP-II. Despite the strengths of the proposed methods, particularly PSLGP-I, NSGP-II remains a robust baseline, especially in high-utilization scenarios, where it demonstrates resilience and strong HV and IGD performance. However, it shows the weakest GD performance among all methods across both high and low-utilization scenarios. Overall, this analysis highlights the effectiveness of PSLGP-I in low-utilization MO-DFJSS scenarios and emphasizes the importance of selecting the most suitable method based on the utilization conditions of the MO-DFJSS problem.

B. Ablation Study

This section conducts an ablation study to analyze the contribution of main components (i.e., multi-criteria selection,

TABLE VII: The test **HV** performance of the proposed method and its variants across ten DFJSS scenarios.

S[1]	PSLGP-I	PSLGP-Ib	PSLGP-Im
1	0.962(0.018)	0.937(0.021)(↓)	0.939(0.022)(↓)
2	0.959(0.012)	0.937(0.020)(↓)	0.932(0.022)(↓)
3	0.909(0.026)	0.860(0.056)(↓)	0.821(0.065)(↓)
4	0.913(0.059)	0.889(0.041)(↓)	0.873(0.036)(↓)
5	0.874(0.055)	0.858(0.032)(↓)	0.834(0.030)(↓)
6	0.937(0.026)	0.917(0.027)(↓)	0.862(0.130)(↓)
7	0.950(0.015)	0.910(0.059)(↓)	0.913(0.083)(↓)
8	0.964(0.017)	0.942(0.026)(↓)	0.927(0.055)(↓)
9	0.960(0.014)	0.944(0.021)(↓)	0.900(0.068)(↓)
10	0.955(0.017)	0.952(0.016)(=)	0.921(0.036)(↓)
w|d|l	-	0|1|9	0|0|10
r	1.2	2.4	2.4

TABLE VIII: The test **IGD** performance of the proposed method and its variants across ten DFJSS scenarios.

S[1]	PSLGP-I	PSLGP-Ib	PSLGP-Im
1	0.022(0.010)	0.037(0.013)(↓)	0.036(0.018)(↓)
2	0.023(0.005)	0.037(0.013)(↓)	0.044(0.019)(↓)
3	0.078(0.015)	0.088(0.035)(=)	0.093(0.041)(=)
4	0.067(0.051)	0.076(0.036)(↓)	0.076(0.024)(↓)
5	0.078(0.050)	0.082(0.023)(↓)	0.093(0.020)(↓)
6	0.047(0.014)	0.059(0.017)(↓)	0.101(0.122)(↓)
7	0.033(0.006)	0.059(0.051)(↓)	0.055(0.076)(↓)
8	0.020(0.012)	0.035(0.021)(↓)	0.053(0.052)(↓)
9	0.025(0.007)	0.034(0.013)(↓)	0.072(0.063)(↓)
10	0.025(0.012)	0.027(0.010)(↓)	0.052(0.032)(↓)
w|d|l	-	0|1|9	0|1|9
r	1.3	2.7	2.0

as well as brood recombination and preselection) to the algorithm's overall performance. By systematically removing or modifying each component, we observe its impact on key performance metrics. For the ablation variants, PSLGP-Im represents PSLGP-I with the **m**ulti-criteria selection process removed, using instead the non-dominated rank and crowded distance selection similar to NSGP-II. PSLGP-Ib removes the **b**rood recombination and preselection process, relying solely on the same population size of generated offspring. Tables VII, VIII, and IX present the results of the proposed PSLGP-I method and its variants across ten DFJSS scenarios, with each table focusing on a different metric: HV, IGD, and GD, respectively. Also, Wilcoxon tests [52] and Friedman tests [53] are performed to statistically compare and rank these methods.

In terms of HV performance (as shown in Table VII), in each scenario, PSLGP-I consistently achieves the highest (best) HV values compared to both PSLGP-Ib and PSLGP-Im. The HV values for PSLGP-Ib are lower than PSLGP-I in all other scenarios, indicating that the removal of the brood recombination and preselection strategy weakens the method's ability to achieve high performance across diverse DFJSS problems. This suggests that brood recombination and preselection strategy enhances the search space exploration and improves the overall scheduling efficiency of the algorithm. PSLGP-I also outperforms PSLGP-Im in all 10 scenarios, indicating that the multi-criteria selection strategy plays a crucial role in enhancing the algorithm's robustness. The win-draw-loss record further supports the dominance of PSLGP-I, with PSLGP-Ib losing in 9 cases and PSLGP-Im in all the 10 cases. In addition, both PSLGP-Ib and PSLGP-Im have an average rank of 2.4, while PSLGP-I ranks highest with 1.2, indicating its consistent top performance.

In terms of IGD performance (as shown in Table VIII), PSLGP-I achieves the lowest (best) IGD values in nearly all scenarios, indicating superior performance over both PSLGP-Ib and PSLGP-Im. When comparing PSLGP-I to PSLGP-Ib, in 9 out of 10 scenarios, PSLGP-Ib exhibits higher (worse) IGD values compared to PSLGP-I. These results suggest that the absence of the brood recombination and preselection strategy reduces PSLGP-Ib's ability to generate solutions close to the optimal front, emphasising the role of this strategy in improving solution quality and diversity. In comparison with PSLGP-Im, PSLGP-I outperforms PSLGP-Im in 9 out of 10 scenarios. PSLGP-Im generally has the highest IGD values among the three methods. This implies that without multi-criteria selection, the algorithm might struggle to balance objectives effectively, resulting in less optimal solutions. The win-draw-loss records confirm PSLGP-I's dominance, with both PSLGP-Ib and PSLGP-Im losing in 9 scenarios each. The average ranks also highlight PSLGP-I's consistent superiority (rank of 1.3), while PSLGP-Im performs slightly better than PSLGP-Ib overall (ranks of 2.0 and 2.7, respectively), although both perform significantly worse than PSLGP-I.

TABLE IX: The test **GD** performance of the proposed method and its variants across ten DFJSS scenarios.

S[1]	PSLGP-I	PSLGP-Ib	PSLGP-Im
1	0.013(0.010)	0.030(0.013)(↓)	0.022(0.011)(↓)
2	0.013(0.009)	0.028(0.011)(↓)	0.018(0.012)(=)
3	0.025(0.019)	0.056(0.029)(↓)	0.021(0.017)(=)
4	0.030(0.045)	0.057(0.070)(↓)	0.029(0.014)(↓)
5	0.034(0.046)	0.046(0.030)(↓)	0.033(0.018)(↓)
6	0.030(0.019)	0.046(0.025)(↓)	0.069(0.104)(↓)
7	0.016(0.010)	0.047(0.045)(↓)	0.032(0.069)(↓)
8	0.014(0.012)	0.031(0.024)(↓)	0.048(0.062)(↓)
9	0.013(0.007)	0.020(0.009)(↓)	0.038(0.056)(↓)
10	0.019(0.016)	0.034(0.022)(↓)	0.040(0.027)(↓)
w|d|l	-	0|0|10	0|2|8
r	1.4	2.8	1.8

In terms of GD performance (as shown in Table IX), PSLGP-I achieves the lowest (best) GD values in all scenarios, showing that it consistently produces solutions closer to the true Pareto front compared to its two variants. When comparing PSLGP-I to PSLGP-Ib, in all 10 scenarios, PSLGP-Ib has higher (worse) GD values than PSLGP-I. These results suggest that the brood recombination and preselection strategy substantially improves solution quality by enhancing diversity and exploration in the search process. PSLGP-Im also shows higher GD values compared to PSLGP-I in all but two scenarios (scenarios 2 and 3, where it matches PSLGP-I). Although PSLGP-Im generally performs better than PSLGP-Ib, it still lacks the precision of PSLGP-I, particularly in scenarios (e.g., scenarios 6 and 8). This indicates that multi-criteria selection helps PSLGP-I refine its solutions by maintaining a balanced search across objectives. The win-draw-loss record further underscores PSLGP-I's dominance, as it outperforms PSLGP-Ib in all scenarios and PSLGP-Im in 8 out of 10 scenarios. The average rank supports this, with PSLGP-I achieving the best overall ranking (1.4), followed by PSLGP-Im (1.8) and PSLGP-Ib (2.8).

Overall, this ablation study highlights the contributions of both brood recombination and preselection, as well as multi-criteria selection strategies in improving the HV, IGD, and GD performance of PSLGP-I. Removing either of these

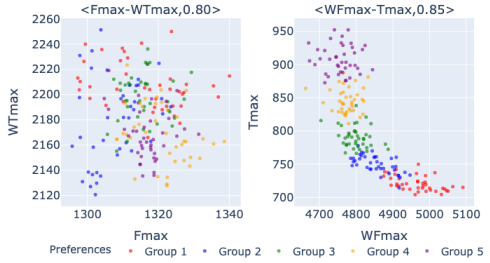

Fig. 8: The solution distribution of the learned heuristic from a single run under varying preferences for scenarios <Fmax-WTmax, 0.80> and <WFmax-Tmax, 0.85>.

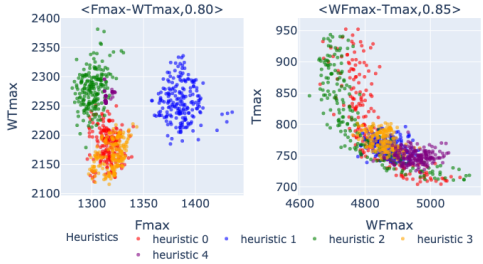

Fig. 9: The solution distribution of the learned multiple heuristics from a single run under varying preferences for scenarios <Fmax-WTmax, 0.80> and <WFmax-Tmax, 0.85>.

elements reduces performance, with the absence of multi-criteria selection (PSLGP-Im) often having the most significant impact. These results show that both strategies contribute to PSLGP-I's effectiveness in generating high-quality solutions for DFJSS, with multi-criteria selection particularly crucial for maintaining a well-balanced and diverse solution set.

VI. Further Analysis

A. Solution Distribution under Different Preferences

This section measures and analyzes the consistency of how preferences are mapped to solutions. Specifically, we aim to verify whether the learned preference-conditioned heuristic can effectively map different preferences to distinct regions of the solution space. To achieve this, we divide the 200 sampled preferences $(\lambda_1, \lambda_2, \ldots, \lambda_{200})$ into five groups, each containing 40 preferences. Specifically, the preferences in group i are represented as $(\lambda_{40\cdot(i-1)+1}, \lambda_{40\cdot(i-1)+2}, \ldots, \lambda_{40\cdot i})$. We then visualize the corresponding test objectives for each group. Fig. 8 gives an example of the solution distribution of the learned heuristic from a single run under varying preferences for the scenarios <Fmax-WTmax, 0.80> and <WFmax-Tmax, 0.85>.

Since the learned heuristic is expected to cover the Pareto front based on preferences, and different preferences should map to specific regions, the ideal distribution would exhibit the following characteristics: (1) Data points of the same color (group) should be clustered together in a relatively small region; (2) Data points of different colors (groups) should occupy distinct regions with minimal overlap; (3) The clusters of different colors should be relatively close to each other, indicating a continuous relationship between the variables. Based on these criteria, as seen in Fig. 8, the left subfigure shows somewhat scattered clusters with some overlap between different colors. There is moderate separation, though overlap persists, and the clusters are relatively close to each other, suggesting a continuous relationship. In the right subfigure, the clusters are more tightly grouped, with minimal overlap. The separation between clusters is more pronounced, and they remain close, maintaining a continuous relationship. Additionally, the shapes of the clusters differ between the two subfigures. The left subfigure features more circular clusters, indicating a uniform distribution, while the right subfigure shows elongated clusters, suggesting a directional relationship—an expected shape.

Overall, while both subfigures display clustering and proximity, the right subfigure demonstrates a more desirable distribution. It better aligns with the criteria of intra-cluster similarity, inter-cluster separation, and overall proximity, suggesting that the data points are more clearly grouped and exhibit a more coherent relationship between the variables. The unexpected result in the left subfigure is primarily due to our proposed multi-criteria selection strategy. A detailed analysis of this result is provided in the Supplementary File.

B. Solution Distribution by Different Heuristics

When learning a preference-conditioned heuristic designed to cover the entire Pareto front by accounting for varying preferences (or directions), the heuristic needs to exhibit strong solution-generation capability across different directions. That is, the heuristic is expected to generate effective solutions in different directions, reflecting varying preferences. However, the effectiveness and solution-generation ability of a heuristic can vary considerably depending on its underlying structure. Analyzing the distribution of solutions generated by different heuristics offers valuable insights into their performance and potential biases. Different from subsection VI-A, which focuses on analyzing the influence of different preferences on the solution regions produced by the best-learned heuristic, this subsection examines the solution regions generated by the top five heuristics.

Fig. 9 illustrates the solution distribution of multiple learned heuristics from a single run under varying preferences for the scenarios <Fmax-WTmax, 0.80> and <WFmax-Tmax, 0.85>. In the solution distribution for scenario <Fmax-WTmax, 0.80> (left subfigure), the clusters corresponding to heuristic 0 (in red) and heuristic 3 (in orange) appear relatively close, with some potential overlap. This suggests that these two heuristics may be generating similar solutions. Conversely, the solutions produced by heuristics 0, 1, and 2 seem more distinct from each other, indicating that these heuristics are focusing on different regions and producing more diverse solutions. Heuristic 4, however, only covers a very small region, demonstrating poor solution-generation ability. In scenario <WFmax-Tmax, 0.85> (right subfigure), the red and green clusters are more spread out, indicating that heuristics 0 and 2 are generating a wider range of solutions. This suggests these heuristics are better at producing distinct solutions under this preference setting, offering strong solution-generation capabilities across different preferences. The green and red clusters' wider spread across preferences indicates that their corresponding heuristics generate solutions with greater diversity, effectively covering a broader range of preferences. Conversely, the more compact

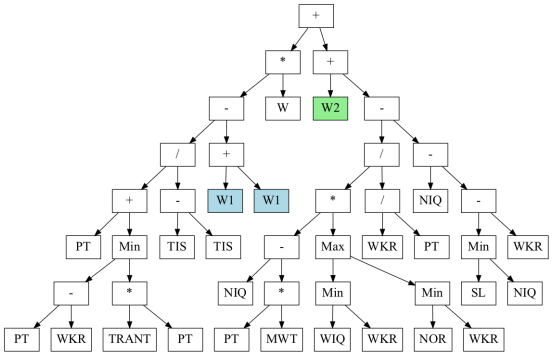

Fig. 10: An example of the sequencing rule from a learned preference-conditioned heuristic from scenario <WFmax-Tmax, 0.85>.

orange, blue, and purple clusters suggest that their heuristics generate more similar solutions, exhibiting limited coverage.

Overall, this analysis demonstrates that the proposed PSLGP method can learn preference-conditioned heuristics that are capable of generating a diverse range of solutions across the Pareto front. However, this outcome is not always guaranteed, as some learned heuristics may focus on only a subregion of the Pareto front. Future work should address this limitation to improve the generation ability of these heuristics.

C. Structure Analysis

This paper presents a novel representation of preference-conditioned scheduling heuristics, incorporating two additional parameters, W1 (w_1) and W2 (w_2), which capture preference-related inputs. This section examines the influence of these parameters on the learned preference-conditioned heuristic. Figs. 10 and 11 illustrate examples of a sequencing rule and a routing rule derived from a preference-conditioned scheduling heuristic generated by the proposed PSLGP-I.

The given sequencing rule (as shown in Fig. 10) can be converted to the following mathematical equation:

$$S_0 = \left(\left(\frac{\text{PT} + \min\left(\text{PT} - \text{WKR}, \text{TRANT} \cdot \text{PT}\right)}{\text{TIS} - \text{TIS}} - 2\text{W1} \right) \cdot W \right) + P_{two} \cdot \left(S_1 - \left(\text{NIQ} - \min\left(\text{SL}, \text{NIQ}\right) \cdot \text{WKR}\right)\right) \tag{7}$$

where:

$$S_1 = \frac{\left(\text{NIQ} - \text{PT} \cdot \text{MWT}\right) \cdot \max\left(\min\left(\text{WIQ}, \text{WKR}\right), \min\left(\text{NOR}, \text{WKR}\right)\right)}{\text{WKR}/\text{PT}} \tag{8}$$

Since $\text{TIS} - \text{TIS} = 0$ and this paper uses a protected division operator (/), $\frac{\text{PT}+\min(\text{PT}-\text{WKR},\text{TRANT}\cdot\text{PT})}{\text{TIS}-\text{TIS}}$ in the first part of the expression will always evaluate to 1 and can therefore be ignored. Given that $\text{W1} + \text{W2} = 1$, we analyze how these parameters affect the behavior of the equation. In terms of the influence of W1, W1 appears twice in the first term as $2 \cdot \text{W1}$. This essentially doubles the influence of W1 in the subtraction part. A higher W1 increases the weight of the negative influence in the first part of the expression, especially when it multiplies W. This reduces the value of the term, emphasizing solutions that avoid configurations where W1 is too large. When it is about the effect of W1 on the overall expression, since W1 is subtracted, a larger W1 will drive the first part of the equation more negatively, pushing the solution to minimize W, which could represent waiting time or a cost factor. Therefore, increasing W1 means the equation will prioritize reducing the penalty associated with this waiting time. In terms of the influence of W2, W2 appears in the second part of the expression, multiplying the more complex part involving NIQ (number in the queue) and other variables like PT (processing time), MWT (machine waiting time), and WKR (work remaining). A higher W2 increases the impact of the term involving the product of queue-related variables, effectively focusing more on optimizing the number in the queue and minimizing its associated penalties. When it is about the effect of W2 on the overall expression, since W2 scales the second term, a larger W2 pushes the solution toward minimizing the impact of NIQ and related variables, possibly at the expense of the penalties from the first part of the equation (controlled by W1). As $\text{W1} + \text{W2} = 1$, increasing one decreases the other, creating a trade-off between their respective influences. A higher W1 means prioritizing reducing penalties related to the work W, while a higher W2 focuses on minimizing penalties associated with queue management and processing times (NIQ, PT, MWT). The optimal balance between W1 and W2 depends on the specific scenario's needs for balancing multiple objectives — whether minimizing work/resource penalties or queue-related delays is more critical.

In summary, in this sequencing rule, W1 controls penalties related to resource/work time, while W2 controls penalties related to queue and processing time variables. A higher W1 will focus on reducing resource-related delays or costs, while a higher W2 prioritizes queue and process optimization. The balance between these factors is crucial to achieving the best trade-off between objectives when considering, depending on the preferences.

The given routing rule (as shown in Fig. 11) is a nested `Max` expression with several arithmetic operations involving variables like WIQ (work in the queue), $TRANT$ (transportation time), W1, W2, etc. The full function is:

$$R_0 = \max\left(R_1, \text{WIQ} + \text{Max}_5 + W \cdot \text{TRANT}\right) \tag{9}$$

where:

$$R_1 = \max\left(\begin{matrix} \max\begin{pmatrix}\text{Max}_1 + (\text{WIQ} + \text{rDD} - (\text{PT} - \text{OWT})) - \text{Max}_2, \\ \text{WIQ} + \text{rDD} - \text{W1} \cdot \text{TRANT}\end{pmatrix}, \\ \max\begin{pmatrix}\text{Max}_3 - (\text{WIQ} - \text{W2} - \text{W1} \cdot \text{TRANT}), \\ \text{Max}_4\end{pmatrix}\end{matrix}\right) \tag{10}$$

$$\text{Max}_1 = \max\left(\text{WIQ} - P_{\text{one}} \cdot \text{TRANT}, W \cdot \text{TRANT}\right) \tag{11}$$

$$\text{Max}_2 = \max\left(\text{NIQ} + \text{OWT}, \text{SL}\right) \tag{12}$$

$$\text{Max}_3 = \max\left(\frac{\text{MWT} \cdot \text{PT}}{\max(\text{PT}, \text{NOR})}, \text{WIQ} - \text{W2}\right) \tag{13}$$

$$\text{Max}_4 = \max\left(\text{NIQ} \cdot (\text{MWT} - \text{MWT}), \frac{\text{TIS}}{\text{WKR}} - \text{W1} \cdot \text{TRANT}\right) \tag{14}$$

$$\text{Max}_5 = \max\left(W \cdot \text{PT}, \frac{W \cdot \text{TRANT}}{\text{TRANT} + \text{W2}}\right) \tag{15}$$

Given that $\text{W1} + \text{W2} = 1$, we can analyze the influence of W1 and W2 on the rule. Both parameters directly affect terms involving TRANT (transportation time). In terms of the influence of W1, W1 appears in subtractions involving TRANT, such as $\text{WIQ} - \text{W1} \cdot \text{TRANT}$ and $\text{WIQ} + \text{rDD} - \text{W1} \cdot \text{TRANT}$. A higher W1 increases the negative impact of transfer time, reducing the overall value of the expression. This makes W1

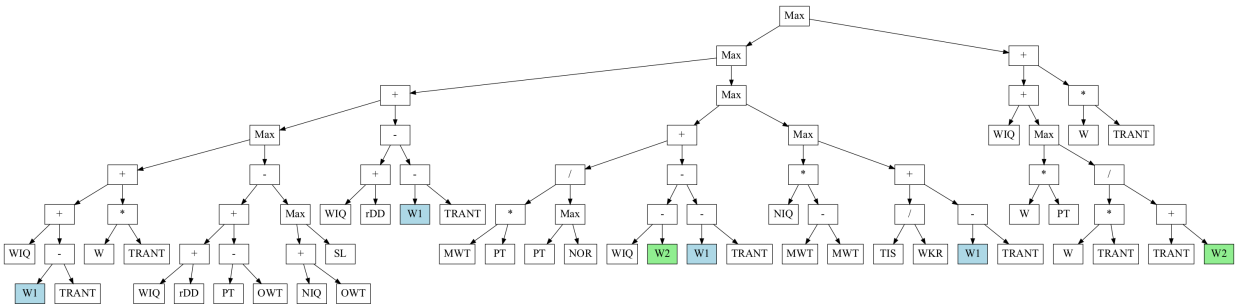

Fig. 11: An example of the routing rule from a learned preference-conditioned heuristic from scenario <WFmax-Tmax, 0.85>.

emphasize solutions that minimize transfer time delays. Also, in the Max_2 function, W1 influences both branches of this Max function by appearing in terms like $\text{W1} \cdot \text{TRANT}$. A higher W1 amplifies the negative influence of transfer time, focusing more on minimizing transfer delays. In terms of the influence of W2, W2 appears in terms like $\text{WIQ} - \text{W2}$, which subtracts from work-in-queue (WIQ). A higher W2 reduces WIQ more strongly, emphasizing solutions where less time is spent waiting in the queue. Since $\text{W1} + \text{W2} = 1$, increasing W1 decreases W2, and vice versa. This trade-off affects the balance between minimizing transfer time (controlled by W1) and minimizing queue time (controlled by W2).

In conclusion, W1 and W2 determine the trade-off between minimizing transfer delays and reducing queue times. A higher W1 prioritizes reducing transportation time (TRANT). A higher W2 focuses on minimizing time spent in the queue (WIQ). The optimal balance between W1 and W2 depends on the specific problem context, but increasing one always reduces the influence of the other due to the constraint $\text{W1} + \text{W2} = 1$, which will influence the rule's ability to handle multiple objectives with different preferences and finally influence the solution location in the Pareto front.

VII. Conclusions

This paper presents a novel Pareto set learning GP framework, designed to learn a single, preference-conditioned scheduling heuristic for MO-DFJSS problems. Unlike conventional methods that require multiple heuristics to cover the Pareto front, this framework effectively adapts to various user preferences within a single heuristic, allowing real-time customization and responsiveness to changing operational conditions. A core innovation of this framework is the preference-conditioned scheduling heuristic representation, which incorporates user preferences as additional inputs, offering a flexible and user-centric approach not seen in traditional heuristics. To maintain efficiency and avoid extended training times, this work introduces a KNN surrogate model for fitness estimation and preselection after brood recombination, allowing it to approximate individual performance across different preferences. Additionally, three novel fitness aggregation strategies are proposed to enhance the heuristic's adaptability, ensuring it aligns with diverse preferences. Experimental results demonstrate that the Pareto set learning GP framework significantly outperforms the baseline multi-objective GP approach, particularly in less busy MO-DFJSS problems, although it faces performance constraints in high-load scenarios. An in-depth analysis of preference impact, solution distribution, and heuristic structure reveals the framework's strength in learning effective preference-conditioned heuristics that align well with user-specified regions of the Pareto front. Nonetheless, challenges remain in achieving consistent alignment across all conditions, especially under high system loads.

Future research could focus on enhancing the robustness of the learned heuristics to ensure that they consistently match diverse preferences and offer comprehensive Pareto front coverage. This could include exploring advanced fitness aggregation techniques, integrating adaptive mechanisms to fine-tune performance in high-load scenarios, and examining hybrid methods that combine preference-conditioned heuristics with other optimization approaches. These advancements would further expand the framework's applicability to a broader range of industrial scheduling challenges.

References

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

[2] W. Yu, L. Zhang, and N. Ge, "An adaptive multiobjective evolutionary algorithm for dynamic multiobjective flexible scheduling problem," *International Journal of Intelligent Systems*, vol. 37, no. 12, pp. 12 335–12 366, 2022.

[3] M. Xu, Y. Mei, F. Zhang, and M. Zhang, "Genetic programming for dynamic flexible job shop scheduling: Evolution with single individuals and ensembles," *IEEE Transactions on Evolutionary Computation*, 2023, DOI:10.1109/TEVC.2023.3334626.

[4] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Survey on genetic programming and machine learning techniques for heuristic design in job shop scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 28, no. 1, pp. 147–167, 2023.

[5] C. Soto, B. Dorronsoro, H. Fraire, L. Cruz Reyes, C. Gomez Santillan, and N. Rangel, "Solving the multi-objective flexible job shop scheduling problem with a novel parallel branch and bound algorithm," *Swarm and Evolutionary Computation*, vol. 53, p. 100632, 2020.

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

[7] M. Xu, F. Zhang, Y. Mei, and M. Zhang, "Genetic programming with multi-case fitness for dynamic flexible job shop scheduling," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2022, pp. 01–08.

[8] C. Zhang, Y. Rao, and P. Li, "An effective hybrid genetic algorithm for the job shop scheduling problem," *The International Journal of Advanced Manufacturing Technology*, vol. 39, pp. 965–974, 2008.

[9] R. Zhang, S. Song, and C. Wu, "A two-stage hybrid particle swarm optimization algorithm for the stochastic job shop scheduling problem," *Knowledge Based Systems*, vol. 27, pp. 393–406, 2012.

[10] L. N. Xing, Y. W. Chen, P. Wang, Q. S. Zhao, and J. Xiong, "A knowledge-based ant colony optimization for flexible job shop scheduling problems," *Applied Soft Computing*, vol. 10, no. 3, pp. 888–896, 2010.

[11] P. Kumar and M. Kumar, "Production scheduling in a job shop environment with consideration of transportation time and shortest processing time dispatching criterion," *International Journal of Advanced Engineering Research and Applications*, 2015.

[12] T. Yang, Z. He, and K. K. Cho, "An effective heuristic method for generalized job shop scheduling with due dates," *Computers & Industrial Engineering*, vol. 26, no. 4, pp. 647–660, 1994.

[13] S. Nguyen, Y. Mei, and M. Zhang, "Genetic programming for production scheduling: a survey with a unified framework," *Complex & Intelligent Systems*, vol. 3, pp. 41–66, 2017.

[14] M. Xu, Y. Mei, F. Zhang, and M. Zhang, "Niching genetic programming to learn actions for deep reinforcement learning in dynamic flexible scheduling," *IEEE Transactions on Evolutionary Computation*, 2024, DOI:10.1109/TEVC.2024.3395699.

[15] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[16] W. B. Langdon and R. Poli, *Foundations of genetic programming*. Springer Science & Business Media, 2013.

[17] Y. Mei, Q. Chen, A. Lensen, B. Xue, and M. Zhang, "Explainable artificial intelligence by genetic programming: A survey," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 3, pp. 621–641, 2022.

[18] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii," in *Proceedings of the International Conference on Parallel Problem Solving from Nature*. Springer, 2000, pp. 849–858.

[19] E. Zitzler, "Spea2: Improving the strength pareto evolutionary algorithm," 2001.

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

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

[22] M. Xu, Y. Mei, F. Zhang, and M. Zhang, "Multi-objective genetic programming based on decomposition on evolving scheduling heuristics for dynamic scheduling," in *Proceedings of the ACM Genetic and Evolutionary Computation Conference Companion*. ACM, jul 2023, pp. 427–430.

[23] M. Wen, R. Lin, H. Wang, Y. Yang, Y. Wen, L. Mai, J. Wang, H. Zhang, and W. Zhang, "Large sequence models for sequential decision-making: a survey," *Frontiers of Computer Science*, vol. 17, no. 6, p. 176349, 2023.

[24] J. P. Huang, L. Gao, X. Y. Li, and C.-J. Zhang, "A cooperative hierarchical deep reinforcement learning based multi-agent method for distributed job shop scheduling problem with random job arrivals," *Computers & Industrial Engineering*, vol. 185, p. 109650, 2023.

[25] X. Lin, Z. Yang, and Q. Zhang, "Pareto set learning for neural multi-objective combinatorial optimization," *arXiv preprint arXiv:2203.15386*, 2022.

[26] X. Lin, Z. Yang, X. Zhang, and Q. Zhang, "Pareto set learning for expensive multi-objective optimization," *Advances in Neural Information Processing Systems*, vol. 35, pp. 19 231–19 247, 2022.

[27] M. Xu, Y. Mei, F. Zhang, and M. Zhang, "Genetic programming and reinforcement learning on learning heuristics for dynamic scheduling: A preliminary comparison," *IEEE Computational Intelligence Magazine*, vol. 19, no. 2, pp. 18–33, 2024.

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

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

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

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

[32] Z. Wu, H. Fan, Y. Sun, and M. Peng, "Efficient multi-objective optimization on dynamic flexible job shop scheduling using deep reinforcement learning approach," *Processes*, vol. 11, no. 7, p. 2018, 2023.

[33] M. Xu, Y. Mei, F. Zhang, and M. Zhang, "A semantic genetic programming approach to evolving heuristics for multi-objective dynamic scheduling," in *Proceedings of the Australasian Joint Conference on Artificial Intelligence (AJCAI)*. Springer, dec 2023, pp. 403–415.

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

[35] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm," *TIK Report*, vol. 103, 2001.

[36] K. Shang and H. Ishibuchi, "A new hypervolume-based evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 5, pp. 839–852, 2020.

[37] Y. Tian, X. Zhang, R. Cheng, and Y. Jin, "A multi-objective evolutionary algorithm based on an enhanced inverted generational distance metric," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2016, pp. 5222–5229.

[38] F. Zhang, Y. Mei, and M. Zhang, "An investigation of terminal settings on multitask multi-objective dynamic flexible job shop scheduling with genetic programming," in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, 2023, pp. 259–262.

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

[40] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Phenotype based surrogate-assisted multi-objective genetic programming with brood recombination for dynamic flexible job shop scheduling," in *Proceedings of the IEEE Symposium Series on Computational Intelligence*, 2022, pp. 1218–1225.

[41] F. Zhang, G. Shi, and Y. Mei, "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*, 2023.

[42] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Multitask multi-objective genetic programming for automated scheduling heuristic learning in dynamic flexible job shop scheduling," *IEEE Transactions on Cybernetics*, aug 2022.

[43] Y. Zeiträg and J. R. Figueira, "Automatically evolving preference-based dispatching rules for multi-objective job shop scheduling," *Journal of Scheduling*, vol. 26, no. 3, pp. 289–314, 2023.

[44] M. Xu, Y. Mei, F. Zhang, and M. Zhang, "Genetic programming with lexicase selection for large-scale dynamic flexible job shop scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 28, no. 5, pp. 1235–1249, 2023.

[45] A. Menchaca-Mendez and C. A. Coello Coello, "Gd-moea: A new multi-objective evolutionary algorithm based on the generational distance indicator," in *Proceedings of the International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 2015, pp. 156–170.

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

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

[48] F. Zhang, Y. Mei, S. Nguyen, K. C. Tan, and M. Zhang, "Instance-rotation-based surrogate in genetic programming with brood recombination for dynamic job-shop scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 5, pp. 1192–1206, 2022.

[49] Y. Zakaria, Y. Zakaria, A. BahaaElDin, and M. Hadhoud, "Niching-based feature selection with multi-tree genetic programming for dynamic flexible job shop scheduling," in *Proceedings of the International Joint Conference on Computational Intelligence*. Springer, 2021, pp. 3–27.

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

[51] I. Das and J. E. Dennis, "Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems," *SIAM journal on optimization*, vol. 8, no. 3, pp. 631–657, 1998.

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

[53] M. R. Sheldon, M. J. Fillyaw, and W. D. Thompson, "The use and interpretation of the friedman test in the analysis of ordinal-scale data in repeated measures designs," *Physiotherapy Research International*, vol. 1, no. 4, pp. 221–228, 1996.