Article

Automatic Design of Energy-Efficient Dispatching Rules for Multi-Objective Dynamic Flexible Job Shop Scheduling Based on Dual Feature Weight Sets

Binzi Xu [1,*], Kai Xu [1], Baolin Fei [1], Dengchao Huang [1], Liang Tao [1] and Yan Wang [2]

[1] School of Electrical Engineering, Anhui Polytechnic University, Wuhu 241000, China; 2230342129@stu.ahpu.edu.cn (K.X.); 2210320117@stu.ahpu.edu.cn (B.F.); huangdengchao@ahpu.edu.cn (D.H.); taoliang@ahpu.edu.cn (L.T.)

[2] School of IoT and Engineering, Jiangnan University, Wuxi 214122, China; wangyan88@jiangnan.edu.cn

* Correspondence: xubinzi@ahpu.edu.cn

Abstract: Considering the requirements of the actual production scheduling process, the utilization of the genetic programming hyper-heuristic (GPHH) approach to automatically design dispatching rules (DRs) has recently emerged as a popular optimization approach. However, the decision objects and decision environments for routing and sequencing decisions are different in the dynamic flexible job shop scheduling problem (DFJSSP), leading to different required feature information. Traditional algorithms that allow these two types of scheduling decisions to share one common feature set are not conducive to the further optimization of the evolved DRs, but instead introduce redundant and unnecessary search attempts for algorithm optimization. To address this, some related studies have focused on customizing the feature sets for both routing and sequencing decisions through feature selection when solving single-objective problems. While being effective in reducing the search space, the selected feature sets also diminish the diversity of the obtained DRs, ultimately impacting the optimization performance. Consequently, this paper proposes an improved GPHH with dual feature weight sets for the multi-objective energy-efficient DFJSSP, which includes two novel feature weight measures and one novel hybrid population adjustment strategy. Instead of selecting suitable features, the proposed algorithm assigns appropriate weights to the features based on their multi-objective contribution, which could provide directional guidance to the GPHH while ensuring the search space. Experimental results demonstrate that, compared to existing studies, the proposed algorithm can significantly enhance the optimization performance and interpretability of energy-efficient DRs.

Keywords: dynamic flexible job shop; genetic programming; dispatching rule; dual feature weight sets; energy-efficient

MSC: 90-08

check for updates

Citation: Xu, B.; Xu, K.; Fei, B.; Huang, D.; Tao, L.; Wang, Y. Automatic Design of Energy-Efficient Dispatching Rules for Multi-Objective Dynamic Flexible Job Shop Scheduling Based on Dual Feature Weight Sets. *Mathematics* **2024**, *12*, 1463. https://doi.org/10.3390/math12101463

Academic Editor: Javier Alcaraz

Received: 16 April 2024
Revised: 7 May 2024
Accepted: 7 May 2024
Published: 9 May 2024

1. Introduction

With the vigorous development of manufacturing technology and artificial intelligence technology, intelligent manufacturing has become one of the mainstream directions of the manufacturing industry [1,2]. So far, artificial intelligence technology has been effectively applied in many directions and fields of the manufacturing industry. In terms of the operation research and management aspect, it mainly focuses on using intelligent optimization algorithms to solve the job shop scheduling problem (JSSP) [3].

JSSP entails the efficient allocation of production resources for a decomposable processing task under certain constraints to optimize performance metrics, such as total processing time, flow time, and tardiness [4]. The flexible job shop scheduling problem (FJSSP) allows for more flexible use of machine resources compared to the JSSP, where each operation of a job can be processed on multiple candidate machines, better reflecting real-world situations [5–7]. Therefore, it is essential to first assign jobs to suitable machines (i.e., the routing decisions) in the FJSSP and then sequence the jobs in the waiting queue of each machine for processing (i.e., the sequencing decisions).

The JSSP and the FJSSP, as typical NP-hard problems, have garnered widespread attention and thorough research from experts and scholars due to their extensive application and complexity in the manufacturing domain. For small-scale problems, researchers tend to utilize exact methods, such as mathematical programming [8] and branch-and-bound [9]. These methods can solve the problem effectively while ensuring the accuracy and optimality of the obtained solutions, but they also suffer from the drawbacks of being computationally intensive and time-consuming. On the other hand, to tackle large-scale problems and tighter time constraints, meta-heuristic algorithms have been viewed as prominent choices. Meta-heuristic algorithms, such as the genetic algorithm [10], the Tabu Search algorithm [11], particle swarm optimization [12], teaching–learning-based optimization [13], and simulated annealing [14], etc., can find approximate optimal solutions in a relatively short time, offering acceptable solutions for decision-making in practical production scenarios.

However, the actual manufacturing environment is usually dynamic, which means that unexpected random events (such as the arrival of new jobs, machine failures, reworking of jobs, etc.) often occur in the dynamic flexible job shop scheduling problem (DFJSSP) [15]. Traditional meta-heuristic methods typically use rescheduling mechanisms to tackle these dynamic events, but these approaches present challenges in terms of increased computational complexity and stability issues [16,17].

As another promising approach, dispatching rules (DRs) demonstrate their effectiveness as a heuristic strategy for solving the DFJSSP due to their scalability, reusability, and rapid response to dynamic events [18–20]. In essence, DRs are priority functions that consist of features containing job shop-related information and mathematical operators. At each routing and sequencing decision point, DRs compute priority values for each scheduling object (such as waiting operations and candidate machines) and choose the most prioritized one for the next processing action. In the past few decades, numerous types of features (e.g., processing time of each operation, idle time of each machine, etc.) and manually designed DRs (e.g., first in, first out, etc.) have been proposed to address various job shop scenarios and optimization objectives [21].

However, it is exceedingly challenging and perhaps even infeasible to depend solely on workers' expertise to clarify potential correlations among different features and to design effective DRs. This is due to the intricate interactive interconnection of various production components in the job shop, as well as the distinct decision environment and information requirements associated with different scheduling decisions in the DFJSSP. In this case, the application of a genetic programming hyper-heuristic (GPHH) for the automated design and generation of appropriate DRs has become a widely used method for effectively addressing the DFJSSP. Numerous studies have demonstrated that using GPHH could automatically evolve DRs that outperform manually designed ones [22,23].

GPHH has the ability to autonomously generate efficient DRs without extensive domain-specific knowledge. This is achieved by continuously adapting the combination of features and mathematical operators in the DRs during evolutionary iterations with predetermined algorithmic parameters (e.g., maximum depth of the tree, function set, feature set, etc.) and genetic operators (e.g., crossover, mutation, etc.). In comparison to most meta-heuristic algorithms for solving the JSSP, GPHH has the advantages of flexible encoding representation, powerful search capability, and easier application to real-world environments. Hence, DRs produced by GPHH are naturally suitable for solving the large-scale DFJSSP. In solving the JSSP, GPHH first uses a set of features (corresponding to leaf nodes) representing the states of jobs, machines, and job shops, as well as a set of mathematical functions (corresponding to non-leaf nodes) to generate DRs automatically and iteratively through off-line training. When scheduling online, the generated DRs are directly used to make scheduling decisions without repeated training by GPHH, thus realizing real-time scheduling [24].

Previous works have shown that the maximum depth of DRs, the function set, and the feature set were the three main factors that could determine the diversity of DRs and the search space of GPHH [25]. If the maximum depth of the DRs is d, and the DRs are generated using the "full method" (i.e., full growth method), the size of the GPHH search space is $|\mathcal{F}|^{2^d-1} \times |\mathcal{T}|^{2^d}$, where $\mathcal{F}$ and $\mathcal{T}$ denote the function set and feature set, respectively. However, the number of effective DRs is much smaller than the above theoretical value, which indicates that finding solutions through GPHH is still a difficult task.

Therefore, reducing the number of features is a viable approach to narrow the search space of GPHH and enhance its searching efficiency [26,27]. The feature set of GPHH can encompass numerous features describing various aspects and forms of information pertaining to the job shop scheduling process, including system-related, machine-related, and job-related features. However, not all of these features contribute positively to scheduling decisions. For instance, the due date of a job is usually considered to be an irrelevant feature for optimizing the mean flow time [26]. In addition, there are correlations between many features, which can cause some of the same information to be double-counted. Therefore, by integrating feature selection techniques with GPHH, it is possible to eliminate irrelevant and redundant features, allowing the algorithm to purposefully explore regions containing more promising DRs and then improve algorithm efficiency.

As far as we know, the research on feature selection methods for GPHH and its variations is still limited and primarily focuses on single-objective problems. The initial studies emphasized measuring the importance of features by calculating the frequency of their occurrence in the best DRs [28]. Mei et al. [29] were the first to propose a metric to measure the feature importance based on the contribution of features to the fitness values of individuals. Compared to feature frequency, this metric could avoid the negative influence brought by redundant features and offer better accuracy. However, these methods were mainly explored in the context of the DFJSSP, considering only sequencing rules. Zhang et al. [30] were pioneers in applying feature selection methods to the DFJSSP, involving two feature sets for both routing and sequencing decisions, respectively. Additionally, in their subsequent work [31], they further developed a continuous two-stage GPHH framework to fully utilize excellent individuals during the feature selection process. However, although these methods could reduce the dimensionality of the feature set and enhance the interpretability of generated DRs through feature selection, they did not lead to an improvement in the optimization performance of GPHH.

Although feature selection has been successfully applied to single-objective problems, no feasible methods have been found for the multi-objective DFJSSP. The main challenge is that calculating the feature importance in multi-objective problems takes into account not only the performance indexes themselves, but also the correlation and non-domination relationships among these indexes. As a result, the existing feature importance measures based on fitness values for single-objective problems are no longer applicable, especially considering that the objectives in multi-objective problems are usually interrelated, and a feature may bring positive effects in one objective but may lead to negative effects in another. Therefore, it becomes very difficult to design an effective feature importance measure for multi-objective problems. In addition, the result of feature selection is heavily dependent on manually set parameters. For instance, the condition for features to be selected in the previous studies [29–31] is that the feature weight value is greater than or equal to half of the total weight value. Such a parameter setting is too rigid, lacking not only fault tolerance and flexibility, but also sufficient adaptability for different scheduling problem scenarios.

Aiming at the limitations of the aforementioned feature selection methods, this paper proposes an improved GPHH with dual feature weight sets to automatically design energy-efficient DRs. In this proposed GPHH, two novel feature weight measures are proposed for calculating the feature weights in the multi-objective DFJSSP and forming the dual feature weight sets. Additionally, a novel hybrid population adjustment strategy is also presented to use the obtained dual feature weight sets to guide the algorithm. The main difference between the proposed feature weight measures and the existing feature selection methods is that the proposed feature weight measures do not select the features but rather assign proper weights to the features according to their contribution/importance to multi-objective problems, which serve as the probabilities of the features being selected in subsequent iterations. This way, the influence of manually set parameters on the feature set and the optimization performance of obtained DRs can be avoided, and the directional guidance for the searching process of GPHH can be provided while ensuring the diversity of generated DRs and the search space of GPHH. In this case, more excellent DRs can be obtained. The main contributions and innovations of this paper are as follows:

1. An improved GPHH algorithm based on dual feature weight sets is designed to guide the exploration of GPHH through measured feature weights, thereby improving the searching efficiency of the algorithm and automatically generating more promising and understandable DRs.
2. In order to measure feature weights (i.e., feature importance) more accurately in the multi-objective DFJSSP, two feature weight measures are proposed: one based on the fitness values of DRs and another based on the diversity of the Pareto front. Based on these two feature weight measures, the feature set for GPHH can be separated as dual feature weight sets for routing and sequencing decisions, respectively.
3. In order to use the obtained dual feature weight sets more effectively, a novel hybrid population adjustment strategy is also given in this paper. This strategy can adjust and refine the current population based on the feature weights so that the irrelevant and redundant features can be eliminated.
4. By considering total energy consumption and mean tardiness as two optimization objectives [32,33], the effectiveness of the proposed GPHH is demonstrated on an energy-efficient DFJSSP by comparing them with the existing related algorithms. Additionally, the specific behaviors and associated impacts of the dual feature weight sets in the scheduling process are also comprehensively analyzed.

2. Background

This section gives a brief description of the energy-efficient DFJSSP considered in this paper and the basic idea of GPHH.

2.1. Mathematical Description of the Energy-Efficient DFJSSP

The energy-efficient DFJSSP presents as a typical combinatorial optimization problem [33]. Its mathematical description is as follows. There are m machines $\mathcal{M} = \{M_1, \cdots, M_m\}$ utilized for processing jobs. Each machine has a designated standby power MP_k in this job shop. New jobs $\mathcal{J} = \{J_1, \cdots, J_n\}$ randomly arrive over time, and their related information is allocated after their arrival, including arrival time (at_i), due date (dd_i), and the specified operation sequence $\{P_{i1}, \cdots, P_{in_i}\}$ (where n_i represents the number of operations for processing job J_i). Each operation P_{ij} has a designated candidate machine set $\mathcal{M}_{ij}$, and this operation can be processed by any machine $M_k \in \mathcal{M}_{ij}$. In this case, the corresponding processing time is PT_{ijk}, and the corresponding energy consumption is EC_{ijk}. Considering the real production situations, the DFJSSP must adhere to the following constraints:

1. Before jobs arrive at the shop floor, the job-related information is unknown and therefore not taken into consideration during the current machining process.
2. Jobs can only be processed upon reaching the job shop, and all operations must be processed in the order given. Each operation can be performed on only one machine selected from its candidate machine set.
3. Machines can process only one job/operation at a time, and the process cannot be interrupted. Additionally, they are in a standby state when not processing, and they consume energy with standby power.

In this paper, we simultaneously optimize two production objectives in the DFJSSP: Mean Tardiness (MT) and Total Energy Consumption (TEC) [33,34], whose mathematical expressions are shown in Equations (1) and (2). Here, MT is used to ensure that considered jobs can be finished in time, while TEC responds to the energy consumption of the whole job shop, which mainly consists of two parts: energy consumption for machining and energy consumption for being on standby.

$$\min \quad MT = \frac{1}{n}\sum_{i=1}^{n} \max\{c_i - dd_i, 0\}, \tag{1}$$

$$\min \quad TEC = \sum_{k=1}^{m}\left(c_{\max} - \sum_{i=1}^{n}\sum_{j=1}^{n_i} PT_{ijk}\right)MP_k + \sum_{i=1}^{n}\sum_{j=1}^{n_i}\sum_{k=1}^{m} EC_{ijk}, \tag{2}$$

where c_i is the completion time of the job J_i, and $c_{\max} = \max_{i=\{1,\ldots,n\}} c_i$ is the completion time of the last job.

2.2. Solving the DFJSSP Based on GPHH

As a well-known optimization approach, GPHH has been widely used in all kinds of optimization problems, especially the combinatorial optimization problem represented by the JSSP [7,35]. As a hyper-heuristic method, GPHH does not directly find a specific solution to the problem but iteratively generates a set of heuristic rules to guide the solution generation. In the JSSP, GPHH generates DRs, which are used to make scheduling decisions to generate a specific scheduling scheme for solving the JSSP, rather than the scheduling scheme itself like meta-heuristic algorithms do [36,37]. For solving the energy-efficient DFJSSP mentioned in Section 2.1, the DRs generated by GPHH in this paper contain two rules: a routing rule and a sequencing rule, which are used to handle the routing decisions and sequencing decisions, respectively.

The DRs generated by GPHH are usually represented as a tree-based structure, as shown in Figure 1. Such a tree-based structure can help GPHH perform genetic operators, such as replication, crossover, and mutation, to generate new DRs. Figure 1 illustrates an example of the DR with two rules. The leaf nodes at the bottom of the tree are constants or features that represent job shop-related information, such as PT standing for the processing time of the current operation, WIQ standing for the total processing time of all the operations in the waiting queue of a machine, etc. The non-leaf nodes are connected by function operators such as {+, −, ×, / }, etc.

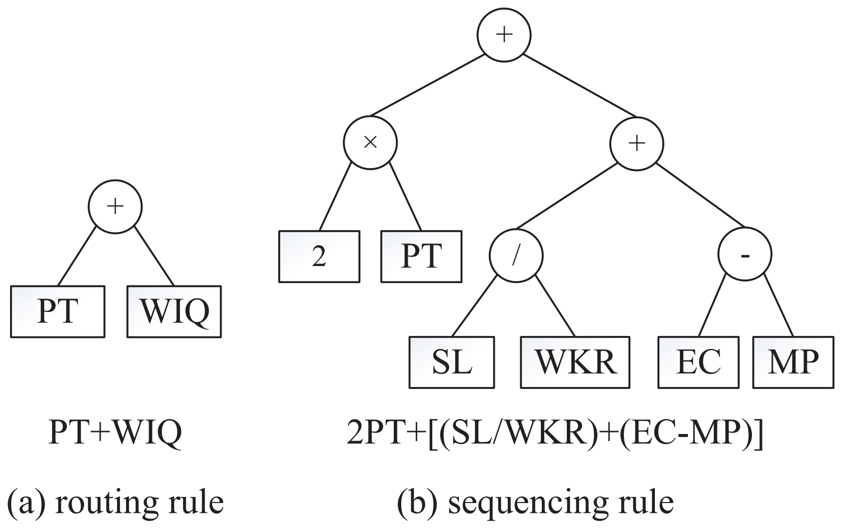

Figure 1. An example of the DR with routing and sequencing rules.

In this paper, only one dynamic event is considered, i.e., random job arrivals, and their related information is unknown until the jobs arrive at the shop floor. A typical scheduling decision-making process for how to use generated DRs to solve the DFJSSP in real time is given in Figure 2. At each decision point, priority values are computed for each scheduling object (i.e., candidate machines for routing decisions or waiting operations in the machines' queues for sequencing decisions) using DRs, and the scheduling object with the minimum priority value is chosen as the decision output. As shown in Figure 2, when a new job arrives, its related information becomes known, and it enters the waiting area. The routing decision, guided by the routing rule, assigns a machine from the corresponding candidate machine set for the current operation of the job. When the machine is machining, the job will be placed into the queue of this chosen machine. The sequencing decision uses the sequencing rule to designate the next job/operation being processed from the machine's queue immediately after it completes one operation. For instance, in the figure, after the operation P_{12}, which means the second operation of job J_1, is processed on machine M_3 based on routing and sequencing decisions, its next operation P_{13} will be assigned to the queue of machine M_2, waiting for machining. This process continues until all operations are completed, leading to the output of the finished job. Routing and sequencing decisions work together to create a scheduling scheme that adheres to constraints and optimizes production objectives.

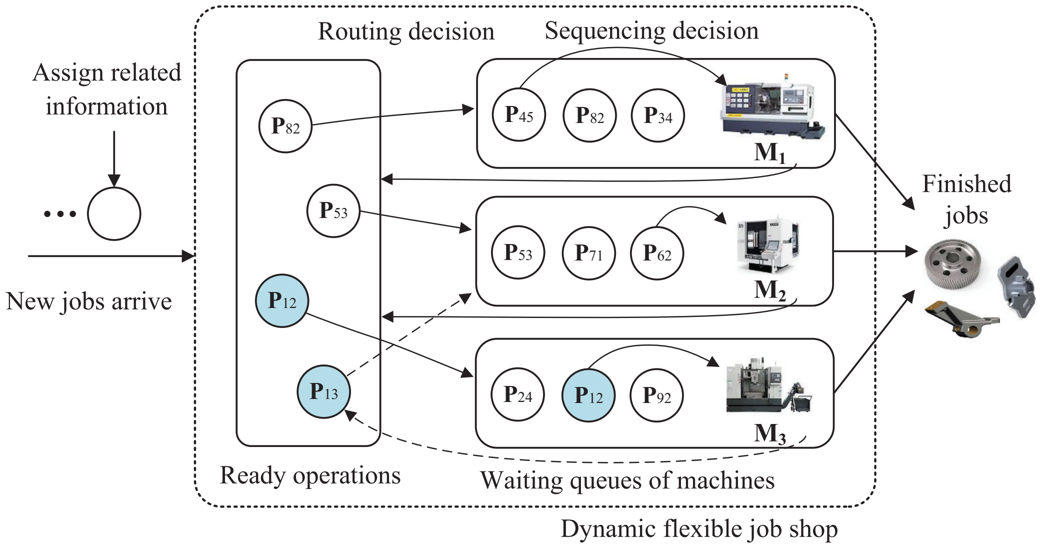

Figure 2. A typical scheduling decision-making process based on DRs.

2.3. *Difference between Feature Selection and Feature Weights*

It has been proven that features in GPHH are not of equal importance [26]. In fact, including irrelevant features in the feature set could reduce the optimization performance of GPHH. Existing feature selection methods for GPHH focus more on using fewer features to obtain more compact and understandable DRs than on improving their effectiveness. For instance, in previous work [31], more interpretable DRs were evolved by applying feature selection methods in GPHH to only select useful features. However, since the results of feature selection are overly dependent on human-set thresholds, it is easy to mistakenly remove both irrelevant and redundant features and features that are useful to the DFJSSP. In this case, GPHH fails to get enough valid information when generating DRs, which ultimately affects the performance of the algorithm.

In addition, different objectives need different features when solving the multi-objective DFJSSP. It is more difficult to measure the contribution and importance of features in such a situation, which further increases the difficulty of feature selection because existing feature selection methods focus mainly on single-objective problems [31].

Therefore, compared to the feature selection methods, this paper proposes two novel feature weight measures for the multi-objective DFJSSP. These measures assign appropriate weight values to features based on their importance and contribution to solving and optimizing multi-objective DFJSSP. In doing so, they can effectively avoid the heavy dependence on human-set thresholds in the selection methods. Additionally, feature weights can provide guidance for the evolutionary process of GPHH while ensuring the algorithm's search space. This characteristic could improve the search efficiency of the algorithm so that higher-quality and more interpretable DRs can be generated.

In summary, the improved GPHH with dual feature weight sets proposed in this paper retains the advantages of the feature selection methods (i.e., improving the search efficiency of GPHH and the interpretability of the generated DRs) while avoiding the negative impact of feature selection on the algorithm performance (i.e., guaranteeing the search space of GPHH).

3. GPHH Based on Dual Feature Weight Sets

3.1. Framework of the Proposed GPHH

This paper considers the diversity of decision information needs for routing and sequencing decisions and introduces the concept of dual feature weight sets into GPHH, which essentially generates two different feature weight sets for the routing and sequencing DRs. For this goal, two novel feature weight measures and a hybrid population adjustment strategy are presented to more effectively utilize the useful information from outstanding individuals during the iterative process, thus producing superior DRs.

Figure 3 depicts the overall framework of our proposed GPHH in detail. Compared with traditional GPHH, Figure 3 highlights the proposed feature weight measure module and feature weight utilization module (including the hybrid population adjustment strategy and the mutation operator in Stage 2) in red.

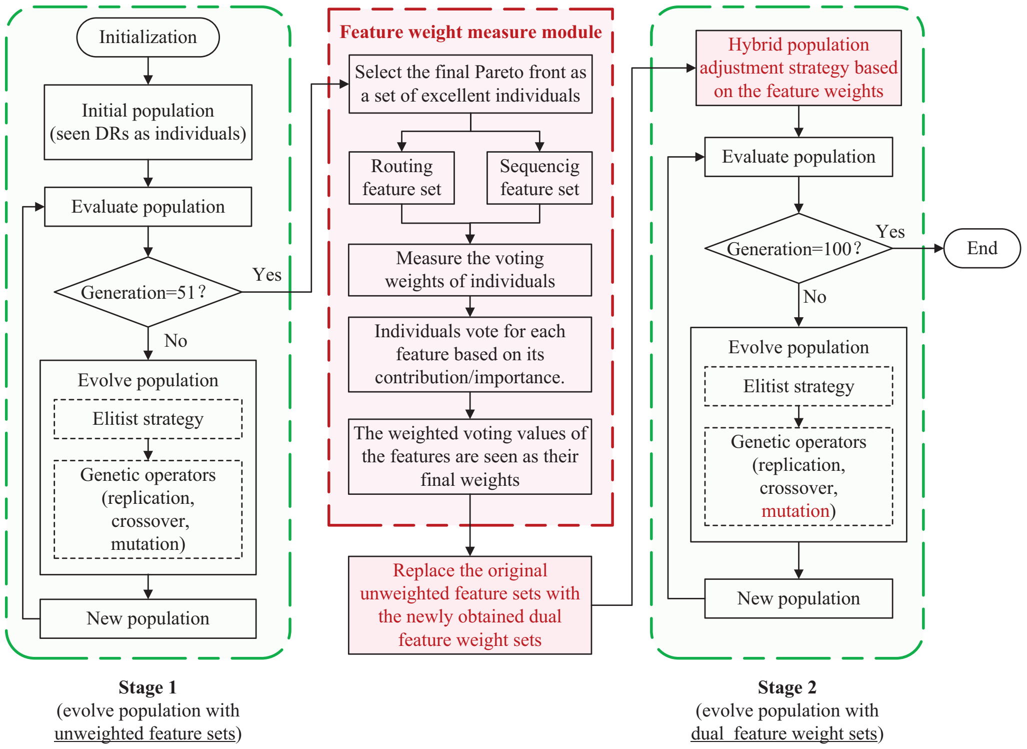
Initialization
Initial population (seen DRs as individuals)
Evaluate population
Generation=51?
Yes
No
Evolve population
Elitist strategy
Genetic operators (replication, crossover, mutation)
New population
Stage 1
(evolve population with unweighted feature sets)
Feature weight measure module
Select the final Pareto front as a set of excellent individuals
Routing feature set
Sequencig feature set
Measure the voting weights of individuals
Individuals vote for each feature based on its contribution/importance.
The weighted voting values of the features are seen as their final weights
Replace the original unweighted feature sets with the newly obtained dual feature weight sets
Hybrid population adjustment strategy based on the feature weights
Evaluate population
Generation=100?
Yes
End
No
Evolve population
Elitist strategy
Genetic operators (replication, crossover, mutation)
New population
Stage 2
(evolve population with dual feature weight sets)

Figure 3. The overall framework of the proposed GPHH with dual feature weight sets.

There are two stages shown in Figure 3. In Stage 1, the algorithm utilizes an unweighted feature set for both routing and sequencing decisions in the initial 50 generations, essentially following the steps of the traditional GPHH. The goal of this stage is to generate a set of high-quality individuals for precise feature weight assessment.

After this, the feature weight measure module calculates the contributions and significance of each feature for two different scheduling decisions, hence forming the routing/sequencing feature weight set (i.e., the dual feature weight sets). Notably, the Pareto front generated from Stage 1 is directly chosen as a set of excellent and diverse individuals in the feature weight measure modules due to the nature of the multi-objective optimization problem. Here, the feature weight measure module only performs one time during the entire iteration of the improved GPHH for the sake of efficiency, as shown in Figure 3.

The resulting new dual feature weight sets are then applied in Stage 2 of our algorithm for 50 additional generations. Specifically, these sets aid in refining the population from Stage 1 by eliminating irrelevant features, which is called population adjustment in Figure 3. Additionally, the dual feature weight sets can also guide the search and evolution direction of the improved algorithm. Features with higher weights, indicating their importance in optimization performance, are more likely to be selected by genetic operators, enhancing the performance of individuals in the subsequent 50 generations of evolution (i.e., Stage 2). Conversely, features with lower weights, having minimal impact on performance, are less likely to be chosen, also leading to the improved optimization performance of individuals. This kind of mechanism can enhance the effectiveness and interpretability of the proposed GPHH without altering the algorithm's search space.

3.2. *Feature Weight Measures for the Multi-Objective DFJSSP*

As shown in Figure 3, accurately measuring the weights of features is crucial for guiding GPHH to generate outstanding DRs. This is because the accuracy of measured feature weights will directly affect the subsequent population adjustment strategy and mutation operator, which in turn affects the search direction of the whole algorithm. For this reason, this paper proposes two different feature weight measures suitable for multi-objective optimization problems: fitness-based and diversity-based feature weight measures.

3.2.1. Fitness-Based Feature Weight Measure

Mei et al. [26] presented a feature selection method for GPHH to tackle the single-objective JSSP and pointed out that the importance of each feature was determined by both individuals' fitness values and the contribution of the features themselves. Following this idea, our study focuses on the non-dominated relationships among individuals in the multi-objective DFJSSP and presents a fitness-based weight measure suitable for solving multi-objective problems.

Algorithm 1 gives the pseudo-code of the proposed fitness-based feature weight measure, which essentially is a weighted voting process. First, the Pareto front $\mathcal{R}$ obtained from the 50th generation (viewed as a group of outstanding individuals) and the unweighted feature sets used in the first 50 generations are utilized as inputs. Then, each feature is evaluated one by one for its contribution to the fitness of the individuals in $\mathcal{R}$. If a feature contributes to the fitness value of that individual, the individual will vote for it. Eventually, each feature accumulates all the obtained voting weights as its own weight value, which is used to measure the importance of this feature. Note that the dual feature weight sets mentioned in this paper indicate that the feature weight sets for routing and sequencing decisions are different.

Equation (3) is used to determine the voting weight $w_{fit}(r)$ of an individual r, where there are m objectives, and $fit_i(r)$ represents the fitness value of the DR r for the ith objective. The goal of this study is to minimize the objective values; hence, the lower the fitness value of an individual, the higher its voting weight.

$$w_{fit}(r) = \sum_{i=1}^{m} \frac{\max\{fit_i(r) \mid r \in \mathcal{R}\} - fit_i(r)}{\max\{fit_i(r) \mid r \in \mathcal{R}\} - \min\{fit_i(r) \mid r \in \mathcal{R}\}} \tag{3}$$

Equation (4) defines the contribution of a feature f to an individual r. Here, when a feature is set to 1, it is analogous to removing this feature from the individual. For instance, (PT + WIQ|PT = 1) = 1 + WIQ. The contribution of a feature to an individual is determined by the result in the numerator of Equation (4). If the final $Con(f, r) > 0$, it indicates that this feature has a positive contribution to the individual, and the individual will vote for this feature.

$$Con(f,r) = \sum_{i=1}^{m} \frac{fit_i(r \mid f=1) - fit_i(r)}{\max\{fit_i(r) \mid r \in \mathcal{R}\} - \min\{fit_i(r) \mid r \in \mathcal{R}\}} \tag{4}$$

Algorithm 1 Fitness-Based Feature Weight Measure

```
Require: the Pareto front $\mathcal{R}$ obtained from the 50th generation and the unweighted feature sets $F_w$
Ensure: the dual feature weight sets $F_w^*$
 1: for all feature $f \in F_w$ do
 2:     $vote(f) \leftarrow 0$
 3:     for all individual/DR $r \in \mathcal{R}$ do
 4:         calculate the voting weight $w_{fit}(r)$ of this individual $r$ using Equation (3)
 5:         calculate the contribution $Con(f,r)$ of this feature $f$ to this individual $r$ using Equation (4)
 6:         if $Con(f,r) > 0$ then
 7:             $vote(f) \leftarrow vote(f) + w_{fit}(r)$
 8:         end if
 9:     end for
10: end for                         ▷ Each feature has been assigned a weight value.
11: $F_w^* \leftarrow$ normalized feature weights for all $f \in F_w$
12: return $F_w^*$
```

3.2.2. Sparsity-Based Feature Weight Measure

Considering that the goal of solving multi-objective problems is to obtain a more widely distributed and diverse Pareto front, in addition to the above proposed fitness-based feature weight measure, this paper also proposes a feature weight measure based on the sparsity of the Pareto front to evaluate the importance/contribution of features. This method pays more attention to the contribution of both features and individuals to the entire Pareto front, whose pseudo-code is given in Algorithm 2.

Algorithm 2 Sparsity-Based Feature Weight Measure

```
Require: the Pareto front $\mathcal{R}$ obtained from the 50th generation and the unweighted feature sets $F_w$
Ensure: the dual feature weight sets $F_w^*$
 1: for all feature $f \in F_w$ do
 2:     $vote(f) \leftarrow 0$
 3:     for all individual/DR $r \in \mathcal{R}$ do
 4:         calculate the voting weight $w_{spa}(r)$ of this individual $r$ using Equation (5)
 5:         set feature $f$ to 1 in individual $r$, denoted as $r \mid f = 1$, and calculate $fit_i(r \mid f = 1)$
 6:         if there exists an individual/DR in $\mathcal{R}$ dominating $r \mid f = 1$, then
 7:             $Con(f,r) = 1$
 8:         else
 9:             $Con(f,r) = 0$
10:         end if
11:         if $Con(f,r) > 0$ then
12:             $vote(f) \leftarrow vote(f) + w_{spar}(r)$
13:         end if
14:     end for
15: end for                         ▷ Each feature has been assigned a weight value.
16: $F_w^* \leftarrow$ normalized feature weights for all $f \in F_w$
17: return $F_w^*$
```

Equation (5) describes the sparsity of an individual as its voting weight, where $fit_i(r+1) - fit_i(r-1)$ represents the difference in fitness values between two individuals adjacent to the individual r. The advantage of this feature weight measure is that it can more evenly assign voting weights to all individuals in the Pareto front, helping to enhance the optimization performance of multi-objective problems considering the non-dominance relationship. Hence, small voting weights are given to individuals that are clustered too close in a very small region, thus ensuring that the final individuals obtained in the Pareto front are uniformly distributed.

$$w_{spa}(r) = \sum_{i=1}^{m} \frac{\mid fit_i(r+1) - fit_i(r-1) \mid}{\max\{fit_i(r) \mid r \in \mathcal{R}\} - \min\{fit_i(r) \mid r \in \mathcal{R}\}} \tag{5}$$

When calculating the contribution $Con(f,r)$ of a feature f to an individual r, the first step is to compute $fit_i(r \mid f=1)$. If $r \mid f=1$ is dominated by any existing individual in the Pareto front, the feature f is considered to have a positive contribution to the individual r. The condition for domination is defined as: for $\forall i \in \{1,2,\cdots,m\}$, $fit_i(r) \leq fit_i(r \mid f=1)$, and there exists at least one $i \in \{1,2,\cdots,m\}$, such that $fit_i(r) < fit_i(r \mid f=1)$.

3.3. Hybrid Population Adjustment Strategy Based on Feature Weights

After completing the measure of feature weights, less important features will be assigned lower weight values. If a feature is given a weight of zero, it can be identified as an irrelevant feature to the problem. Although GPHH itself has the inherent ability to select suitable features during the iteration, it is still challenging to effectively eliminate these irrelevant features solely through genetic operators such as crossover and mutation. Allowing these irrelevant features to persist in individuals will lead to the contamination of the whole population, resulting in more individuals with redundant, duplicated, and unnecessary features. This hinders the further evolution of the population and the optimization performance of generated DRs.

To this end, the existing studies focusing on feature selection methods propose the following three population adjustment strategies [31], which remove unselected features from the current population and improve the interpretability of the resulting DRs based on the selected feature set.

1. Simply replace each unselected feature with a constant of 1. This strategy is proposed based on the above calculation of the feature contribution $Con(f,r)$, which can maximally retain the structure and performance of the current good individuals.
2. Replace unselected features with other selected features. However, such random substitution may change the behavior of good individuals in some aspects, affecting their final optimization performance.
3. Directly use the resulting selected feature set to randomly initialize the population. This way, the effective information of good individuals in the current population will be lost.

Related studies using feature selection methods to solve single-objective problems [31] have shown that the first strategy can better improve the performance of the algorithm compared with the third strategy. In addition, Section 3.2 of this paper also adopts the basic idea of calculating the feature contribution $Con(f,r)$ in the first strategy as the basis of the proposed feature weight measures. Hence, this method not only helps to eliminate redundant information and retain the individuals' structure, but also may improve the performance of these individuals.

Therefore, on the basis of the above existing work, this paper proposes a novel hybrid population adjustment strategy based on the obtained dual feature weight sets. In this hybrid strategy, the first and third strategies are both applied to adjust the current population so that the irrelevant features (whose weights are zero) can be removed and the search efficiency can be enhanced.

Algorithm 3 gives the pseudo-code of this hybrid population adjustment strategy. Specifically, Algorithm 3 only removes irrelevant features from the top 20% of individuals using the first strategy to retain the valid information of good individuals. For the remaining 80% of non-excellent individuals, they are directly deleted and regenerated based on the obtained dual feature weight sets using the third strategy to improve the diversity of individuals in the population while deleting irrelevant features.

Algorithm 3 Hybrid Population Adjustment Strategy Based on Feature Weights

```
Require: the population $\mathcal{P}$ generated from the 50th generation and the obtained dual
    feature weight sets $F_w^*$
Ensure: the new population $\mathcal{P}^*$
 1: sort individuals by the non-dominated rank and crowding distance
 2: for all individual/DR $r \in \mathcal{P}$ do
 3:     if $r$ is the top 20% of all the individuals in $\mathcal{P}$ then
 4:         for all feature $f \in F_w^*$ do
 5:             if $weight(f) < 0.0001$ then
 6:                 replace feature $f$ with a constant of 1
 7:                                                  ▷ This is the first strategy.
 8:             end if
 9:         end for
10:     else
11:         use $F_w^*$ to randomly generate a new individual and replace the old one
12:                                                  ▷ This is the third strategy.
13:     end if
14: end for
15: return $\mathcal{P}^*$
```

4. Experimental Design

4.1. Simulation Model Design

The configuration of the energy-efficient DFJSSP simulation model adopted in this paper is shown in Table 1, which has been extensively employed in previous research [30,33,38].

Table 1. Configuration of the energy-efficient DFJSSP simulation model.

Parameter	Setting
Number of machines m	10
Number of arrived jobs n	2000
Number of warm-up jobs	500
Number of operations per job	discrete $U(1, 10)$
Available machines per operation	discrete $U(1, 10)$
Job arrival process	Poisson process
Utilization level u	0.85, 0.95
Due date factor α	2, 4, 6
Mean processing time $\overline{PT}_{ij}$	discrete $U(1, 99)$
Mean energy consumption $\overline{EC}_{ij}$	discrete $U(1, 99)$
Standby power MP_k	{10, 12.5, 4.5, 3.6, 7.0, 1.5, 8.5, 2.2, 22.9, 6.4}

According to existing research [35,39], the machine utilization level u and the due date factor α are the primary factors defining the workload of different job shop environments. Therefore, this paper considers two machine utilization levels {0.85, 0.95} and three due date factors {2, 4, 6}, resulting in a total of 2 × 3 = 6 different production scenarios. During the training phase, the GPHH reruns independently 30 times under each scenario using different random seeds. To evaluate the effectiveness of the generated DRs, the results of each run undergo 100 independent tests. The average of these 100 test results is then taken as the final fitness value of the obtained DRs, thereby validating their real generality and applicability.

4.2. Parameter Settings of GPHH

Table 2 presents the initial feature set of GPHH in this paper, consisting of 28 features related to the JSSP. These features are commonly utilized in existing research [33] and encompass various aspects such as information related to jobs, machines, and the job shop itself.

Table 2. Feature set of the GPHH.

No.	Feature	Description
1	PT	Processing time PT_{ijk} of operation P_{ij} on M_k
2	SL	Slack of the job J_i
3	OWT	Waiting time of operation P_{ij} since ready
4	NPT	Processing time of the next operation P_{ij+1}
5	WKR	Work remaining for the job J_i
6	TIS	Time of the job J_i in job shop
7	WIQ	Workload in the queue of the machine M_k
8	MWT	Waiting time of the machine M_k since ready
9	DD	Due date of the job J_i
10	MRT	Ready time of the machine M_k
11	ORT	Ready time of the operation P_{ij}
12	WINQ	Workload in the queue of the next machine
13	AT	Arrival time of the job J_i
14	NRT	Ready time of the next machine
15	EC	Energy consumption EC_{ijk} of operation P_{ij} on M_k
16	NEC	Energy consumption of the next operation P_{ij+1}
17	ECR	Energy consumption remaining for the job J_i
18	EIQ	Total energy consumption of all operations in the queue of the next machine M_k
19	EINQ	Total energy consumption of all operations in the queue of machine M_k
20	MP	Standby power of machine M_k
21	NOR	Number of remaining operations of the job J_i
22	NOS	Number of optional machines for the operation P_{ij}
23	NIQ	Number of operations in the queue of machine M_k
24	NINQ	Number of operations in the queue of next machine
25	NOPS	Number of operations of the job J_i
26	RPT	Relative processing time = $\frac{PT_{ijk}}{\min\left\{PT_{ijk}\mid k=1,\ldots,\lvert\mathcal{M}_{ij}\rvert\right\}}$
27	REC	Relative energy consumption = $\frac{EC_{ijk}}{\min\left\{EC_{ijk}\mid k=1,\ldots,\lvert\mathcal{M}_{ij}\rvert\right\}}$
28	RMP	Relative standby power = $\frac{MP_k}{\min\{MP_k\mid k=1,\ldots,m\}}$

The function set includes $\{+, -, \times, /, \max, \min\}$ [27,33], where the division returns one if divided by zero. Additionally, max/min are functions that take two inputs and return their maximum/minimum values, respectively.

Table 3 shows the parameter setting of the GPHH in this section [27,33]. The feature weights are measured at the 51st generation, and this process is executed only once in the whole algorithm.

Table 3. Parameter settings of GPHH.

Parameter	Setting
Initialization	Ramped half-and-half
Population size	600
Maximum depth of DRs	8
Crossover rates	80%
Mutation rates	15%
Reproduction rates	5%
Parent selection	Tournament selection with size 7
Elitism	10 best individuals
Number of generations	101
Feature weight measure	At the 51st generation
Population adjustment strategy	See Algorithm 3

4.3. Comparison Design

In order to effectively analyze the performance of our proposed GPHH with dual feature sets, which considers feature weights during the iterations, three existing algorithms are compared in this section. In this way, a total of five improved GPHH algorithms are compared in this section as follows.

1. GPLWT [40] (i.e., GPHH-LWT) uses the Least Waiting Time (LWT) as its routing rule, and the sequencing rule is generated by GPHH. In other words, this algorithm only automatically evolves sequencing rules via GPHH while its routing rule is fixed. In this section, it is considered as the baseline for solving the energy-efficient DFJSSP.
2. GPDR [33] (i.e., GPHH-Delayed-Routing) is one of the current state-of-the-art algorithms, which adopts both multi-tree representation [39] to generate routing and sequencing rules simultaneously and a delayed routing strategy to ensure the timeliness of the feature information at the decision-making point. In this paper, GPDR is used as the basic algorithm, in which the proposed feature weight measures and the hybrid population adjustment strategy are introduced.
3. GPFS [31] (i.e., GPHH-Feature-Selection) is the GPHH algorithm that incorporates a feature selection method. In their work, GPFS is only used to solve single-objective problems. It is applied to multi-objective problems based on the linear weighting method in this section so that its performance can be analyzed.
4. GPFW(fit) (i.e., GPHH-Feature-Weight based on fitness) is the GPDR with dual feature sets that adopts the feature weight measure based on fitness values of individuals (see Section 3.2.1) and the hybrid population adjustment strategy proposed in Section 3.3.
5. GPFW(spa) (i.e., GPHH-Feature-Weight based on sparsity) is the GPDR with dual feature sets that adopts the feature weight measure based on sparsity of the Pareto front (see Section 3.2.2) and the hybrid population adjustment strategy proposed in Section 3.3.

4.4. Performance Measures for Comparison

When assessing the quality of the Pareto front obtained by the algorithms, this paper employs two commonly used performance metrics for multi-objective optimization problems: Hyper-Volume (*HV*) and Inverted Generational Distance (*IGD*) [41,42]. Their calculation formulas are as follows:

$$HV = \bigcup_{i=1}^{n_{\mathrm{PF}}} v_i, \tag{6}$$

$$IGD = \frac{1}{n_{\mathrm{APF}}}\left(\sum_{i=1}^{n_{\mathrm{APF}}} d_i\right), \tag{7}$$

where n_{PF} is the number of individuals in the Pareto front, v_i is the hyper-volume formed by individuals in the Pareto front and a reference point, n_{APF} is the number of individuals in the true Pareto front, and d_i is the minimum Euclidean distance from individuals in the true Pareto front to the Pareto front obtained by algorithms.

An outstanding algorithm is expected to have a higher *HV* value and a lower *IGD* value, indicating that the Pareto front generated by this algorithm is closer to the true Pareto front and more evenly distributed. Normalization is essential when computing *HV* and *IGD*. As the true Pareto front is unknown in our energy-efficient DFJSSP, the results from all algorithms in this paper are amalgamated, and the resulting Pareto front is utilized as an approximate true Pareto front. *HV* also necessitates a reference point, set as $1 + 1/(n_{\text{APF}} - 1)$ [43], where n_{APF} represents the number of individuals in the approximate true Pareto front.

5. Experimental Results Analysis

5.1. Overall Performance Analysis of the Algorithms

Tables 4 and 5 show the mean and standard deviation of the *HV* and *IGD* values obtained by the different algorithms after 30 independent runs for optimizing both *MT* and *TEC* in six different scenarios. Additionally, Wilcoxon rank-sum tests were conducted (at a 5% confidence level) to further analyze the significance of their optimization performance [44]. Here, GPFW(fit) and GPFW(spa) are compared with three existing algorithms (i.e., GPLWT, GPDR, and GPFS). If the *p*-value of the Wilcoxon rank-sum test is less than 0.05 for all of them, the *significant differences* are marked in bold in the tables.

Table 4. Mean (standard deviation) of *HV* values for different algorithms in six scenarios.

Algorithms	**<0.85–2>**	**<0.85–4>**	**<0.85–6>**	**<0.95–2>**	**<0.95–4>**	**<0.95–6>**
GPLWT	0.324 (0.001)	0.549 (0.000)	0.602 (0.000)	0.392 (0.001)	0.578 (0.001)	0.668 (0.001)
GPDR	0.977 (0.010)	1.000 (0.007)	1.024 (0.005)	0.970 (0.009)	1.005 (0.005)	1.027 (0.006)
GPFS	0.923 (0.101)	0.996 (0.018)	1.024 (0.004)	0.957 (0.033)	0.994 (0.024)	1.017 (0.019)
GPFW(fit)	**0.985 (0.008)**	1.002 (0.005)	1.024 (0.005)	**0.982 (0.010)**	1.011 (0.006)	1.029 (0.005)
GPFW(spa)	**0.984 (0.009)**	1.002 (0.006)	1.024 (0.005)	**0.983 (0.012)**	**1.012 (0.006)**	1.029 (0.005)

Under the Wilcoxon rank-sum test with the significance level of 0.05, the significantly better results are marked in bold.

Table 5. Mean (standard deviation) of *IGD* values for different algorithms in six scenarios.

Algorithms	**<0.85–2>**	**<0.85–4>**	**<0.85–6>**	**<0.95–2>**	**<0.95–4>**	**<0.95–6>**
GPLWT	0.538 (0.001)	0.420 (0.000)	0.398 (0.000)	0.457 (0.001)	0.372 (0.001)	0.325 (0.001)
GPDR	0.014 (0.005)	0.025 (0.006)	0.047 (0.010)	0.025 (0.006)	0.044 (0.013)	0.032 (0.007)
GPFS	0.044 (0.074)	0.025 (0.012)	0.045 (0.010)	0.034 (0.021)	0.045 (0.020)	0.036 (0.012)
GPFW(fit)	**0.009 (0.006)**	0.024 (0.005)	0.046 (0.008)	0.019 (0.006)	**0.037 (0.014)**	0.031 (0.007)
GPFW(spa)	**0.009 (0.005)**	0.023 (0.006)	0.047 (0.008)	**0.018 (0.005)**	0.040 (0.010)	0.030 (0.005)

Under the Wilcoxon rank-sum test with the significance level of 0.05, the significantly better results are marked in bold.

Overall, one can see that GPLWT is the worst among the five algorithms. Of the three existing algorithms, GPDR performs the best. As for the two algorithms proposed in this paper, GPFW(fit) and GPFW(spa) both show better optimization performance when the problem is difficult (see scenarios <0.85–2> and <0.95–2> in Tables 4 and 5).

From these tables, it can be observed that the performance of all algorithms is significantly better than GPLWT, indicating that using routing rules generated by GPHH is more effective than using a fixed LWT rule. This is because GPHH can generate appropriate routing rules based on the current job shop state for routing decisions, thereby creating more rational waiting queues for the machines and enhancing the following sequencing decisions.

Among the three existing GPHH algorithms, GPDR and GPFS are both the current state-of-the-art algorithms that consider the influence of GPHH's feature sets. It is pointed out that GPFS can filter out insignificant features in single-objective problems, making the generated DRs more interpretable and similar performance-wise to traditional GPHH by utilizing fewer features [31]. In this section, GPFS is applied to multi-objective problems using the linear weighting method. The results shown in Tables 4 and 5 indicate that the performance of GPFS is inferior to GPDR, with a notably higher standard deviation in most scenarios, especially in situations with a tighter due date factor such as <0.85–2> and <0.95–2>. This suggests that a simple transfer of the GPFS to multi-objective problems through the linear weighting method is not appropriate. The selected features cannot satisfy the requirements of multi-objective optimization well, thus impacting the algorithm's optimization performance and stability.

When comparing the two algorithms proposed in this paper with three existing algorithms, especially GPDR, it can be seen that the proposed algorithms are better than the other algorithms overall, especially in the scenarios of <0.85–2> and <0.95–2>, which show significant advantages. This phenomenon suggests that the proposed fitness-based and sparsity-based feature weight measures can more accurately calculate the importance/contribution of each feature to solving the multi-objective optimization problem, which further improves the algorithm's search efficiency and optimization performance. Tables 4 and 5 also show that the proposed algorithm has limited performance improvement relative to the basic algorithm (i.e., GPDR), especially in the case of easier problems. This is due to the fact that the GPHH itself has a certain function of feature selection, because the performance of the DRs that incorporate irrelevant and non-essential features is not good, and thus it will be eliminated by the GPHH in the process of iterative evolution. Therefore, even with the introduction of the dual feature weight sets, the improved GPFWs (i.e., GPFW(fit) and GPFW(spa)) will not have a significantly larger performance improvement.

The comparison between GPFW(fit) and GPFW(spa) reveals that their results are very close to each other, with no statistically significant differences. In theory, the sparsity-based feature weight measure may be slightly better than the fitness-based method, as the former takes into account the non-dominated relationships between individuals. This suggests that the Pareto front obtained after 50 iterations of GPHH is good enough to be used to accurately measure the feature weights.

5.2. Comparison of Training and Testing Time

Training and testing time are two important measures of the efficiency of an algorithm. Hence, the average training time for 30 independent runs of all five algorithms is given in Table 6. One can see that GPLWT has the shortest training time because it uses a fixed routing rule (i.e., LWT) and is less computationally intensive. Although GPFS, GPFW(fit), and GPFW(spa) all need extra steps to select features or measure feature weights, there is no significant difference between these algorithms and GPDR. This indicates that the introduction of the feature selection module or feature weight measure module does not bring too much additional computation cost and time to the algorithms, and the overall training time consumed is still acceptable.

Table 6. Mean training time of 30 independent runs for all five algorithms (s).

Algorithms	<0.85–2>	<0.85–4>	<0.85–6>	<0.95–2>	<0.95–4>	<0.95–6>
GPLWT	5124	4988	5032	4587	4602	4495
GPDR	11,745	11,568	11,814	10,656	11,948	12,530
GPFS	12,670	11,923	11,086	9657	9502	10,183
GPFW(fit)	13,029	12,383	13,441	10,122	10,590	10,097
GPFW(spa)	10,741	10,375	13,697	11,638	13,791	12,864

Table 7 gives the average testing time of the DRs obtained by different algorithms after testing. It can be observed that these DRs take only about 20 ms (or less) to complete all the scheduling decisions in one single simulation. In this case, the decision time spent by these DRs will be even less for each decision point, thus meeting the requirement of real-time decision-making.

Table 7. Mean testing time for DRs obtained by different algorithms (ms).

Algorithms	<0.85–2>	<0.85–4>	<0.85–6>	<0.95–2>	<0.95–4>	<0.95–6>
GPLWT	3.90	3.00	3.00	4.40	4.04	3.43
GPDR	14.36	15.58	16.17	14.34	16.93	16.50
GPFS	11.31	15.08	13.26	12.99	13.74	13.95
GPFW(fit)	21.00	20.95	23.02	19.45	22.10	25.04
GPFW(spa)	15.03	15.90	14.45	14.88	16.73	17.42

6. Behavior Analysis of the Proposed Algorithms

6.1. Analysis of the Number of Unique Features

The number of unique features refers to the quantity of different types of features included in the DRs [31,45]. In this study, the quantity of unique features existing in the final population is analyzed, which should positively contribute to the performance of the proposed algorithms. The specific calculation involves examining the number of different types of features present in the final population. A lower number of unique features indicates that the DRs formed by these features are more concise, possess better interpretability, and are consequently more comprehensible to humans.

Figure 4 illustrates the mean and standard deviation of the number of unique features in the DRs generated by different algorithms over six scenarios. Since GPLWT only evolves sequencing rules, it is excluded from the analysis in this section. In this context, GPFS demonstrates a relatively lower number of unique features due to its aim of generating high-quality DRs using a smaller feature set. The number of unique features of GPFS is mainly determined by the threshold value set during the feature selection, where features are selected if their voting weight is greater than half of the total voting weight [31]. Consequently, features with weights below half of the total voting weight are removed, leading to a significant reduction in the number of feature types. However, this reduction also results in a decrease in the optimization performance of the algorithm.

In contrast, our proposed feature weight measures do not directly remove features like GPFS does. Only irrelevant features with a weight of zero are completely removed in the hybrid population adjustment strategy. Additionally, the obtained dual feature weight sets can guide the evolutionary direction of GPHH, as less important features may not appear in the final population. This also contributes to a reduction in the number of unique features. Therefore, the results in Figure 4 suggest that the proposed GPFWs with dual feature weight sets can effectively decrease the number of unique features in generated DRs and even improve the performance of GPHH. This reduction contributes to the evolution of more readable and interpretable energy-efficient DRs. Furthermore, it underscores the rationale and effectiveness of introducing feature weights into the GPHH, since the algorithm does benefit a lot, even in complex multi-objective environments.

Additionally, it can be observed from Figure 4 that the number of unique features for sequencing rules is optimized more easily. One possible reason is that sequencing decisions require relatively less decision information. Sequencing decisions are typically more concerned with job/operation-related information, since the scheduling objects of a sequencing decision are the job/operations in the machines' queues. In contrast, routing rules need to determine which candidate machine is suitable for processing the current job/operation, taking into account the availability of the machine, the status of the processing queue, and other factors in addition to the characteristics of the job/operation itself. This makes the routing rules require more complex decision information (i.e., a greater number of unique features) to assist in decision-making. Hence, the number of unique features for routing rules is relatively higher, making more complex routing decisions.

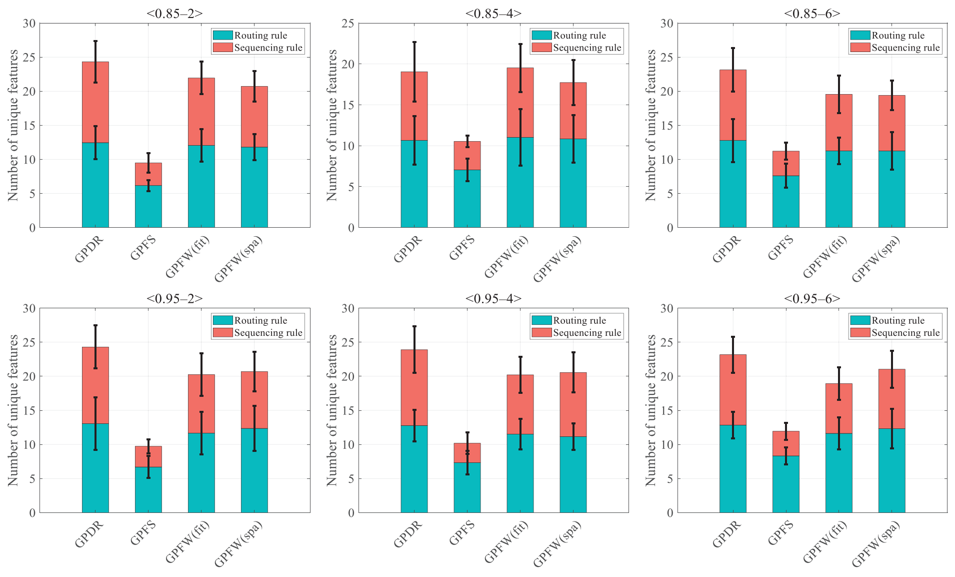

Figure 4. The numbers of unique features of both routing and sequencing rules obtained by different algorithms.

6.2. Analysis of the Rule Sizes of DRs

Rule size is another key indicator that describes the readability and interpretability of DRs. Typically, the smaller the rule size, the lower the depth of the rule, the simpler the structure of the rule, and the easier it is understand.

Figure 5 gives the rule sizes of the DRs obtained by each algorithm. Overall, the rule sizes of DRS generated by these algorithms are not significantly different. Through the introduction of feature selection methods or feature weight measures, the rule sizes of DRs obtained from GPFS and the proposed GPFWs are not significantly reduced, and they only show advantages in a few scenarios, e.g., <0.95–6>.

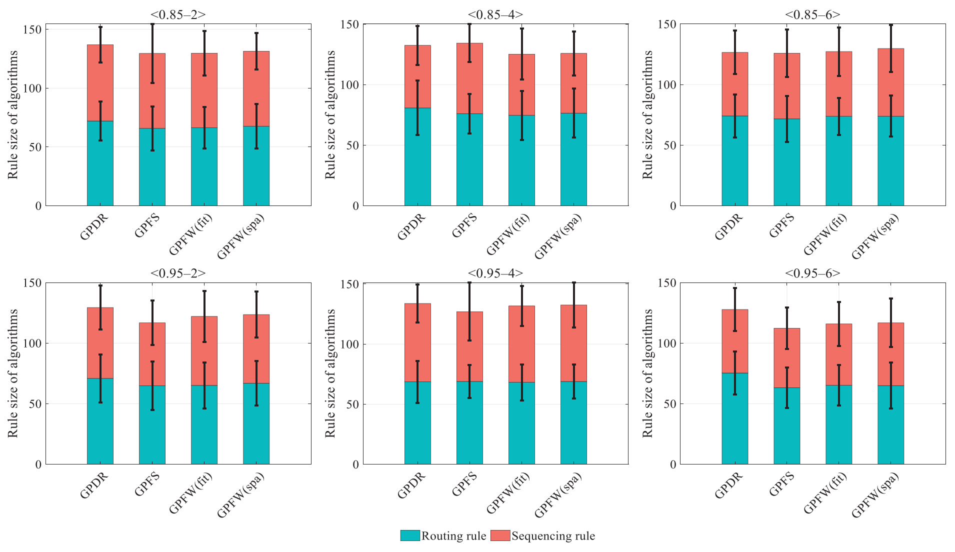

Figure 5. The rule sizes of both routing and sequencing rules obtained by different algorithms.

This result is also expected because what really reduces the rule size is the initialization method (i.e., the third strategy) within the population adjustment strategy. The top 20% of outstanding individuals, although inheriting the valid information from the first 50 generations, also have relatively large rule sizes and are more likely to be selected by genetic operators in the next 50 generations, thus leading to a rise in the rule size of other individuals in the population.

6.3. Conjoint Analysis of Feature Weights and Feature Frequencies

In order to gain insight into the importance/contribution of different features in the evolutionary process of GPHH, this section takes the examples of GPFW(fit) and GPFW(spa) to conduct a conjoint analysis of feature weights and feature frequencies. Figures 6 and 7 present the feature weights and frequencies of routing/sequencing rules, respectively, in six scenarios using these two algorithms, and the statistics are the average of the results of 30 independent runs.

1. Gen50 counts the frequency of each feature in the individuals from the Pareto front that GPFW(fit) and GPFW(spa) have iterated to the 50th generation. Because the first 50 generations are the same for these two algorithms, the same results are obtained.
2. Gen51(fit) and Gen51(spa) denote the feature weights obtained by the fitness-based and sparsity-based feature weight measures, respectively, at generation 51.
3. Gen100(fit) and Gen100(spa) denote the frequency of each feature in the individuals from the final obtained Pareto front after 100 generations of GPFW(fit) and GPFW(spa), respectively.

Figures 6 and 7 unambiguously demonstrate that the results of Gen50, Gen51(fit), Gen51(spa), Gen100(fit), and Gen100(spa) are generally congruent, indicating the correlation between feature weights and feature frequencies. The feature frequencies in Gen50 align well with the feature weights obtained in Gen51(fit) and Gen51(spa), suggesting that feature frequencies can also, to some extent, measure the importance of features. On the other hand, this indirectly implies that GPHH itself has the feature selection ability. The Pareto front achieved after 50 generations is excellent enough to serve as a set of outstanding individuals for feature weight measures.

The minor difference between the results of Gen51(fit) and Gen51(spa) suggests a strong correlation between them, offering diverse viewpoints on the importance of features in solving multi-objective problems. This conclusion is consistent with the findings in Tables 4 and 5, suggesting that, when the results of feature weight measures are similar, the final optimization performance also tends to be similar.

When contrasted with the feature frequencies in Gen50, the results in Gen100(fit) and Gen100(spa) show that features with higher weights have higher frequencies, while lower-weighted features have lower frequencies. This implies that the obtained dual feature weight sets indeed guide the algorithm's search direction in the latter 50 generations. The proposed hybrid population adjustment strategy based on feature weights aids the algorithm in mitigating the negative effects of irrelevant features.

Another observation from Figures 6 and 7 is that the results of most features' weights and frequencies are quite similar in all scenarios. Specifically, features such as PT, MP, and their respective related features RPT, RMP, and WIQ, MWT, NIQ, etc., demonstrate a high level of importance in the generation of routing rules. This aligns with the actual situation, as PT, MP, RPT, and RMP play critical roles in optimizing both MT and TEC. Combining them with features like WIQ, MWT, and NIQ enables a more effective and rational allocation of machine resources, leading to increased speed in job processing. This also ensures that jobs are scheduled on machines with higher MP, thereby reducing their energy consumption from being idle.

From the perspective of sequencing rules, PT, SL, and WKR are three important features. Previous studies [26,27] have demonstrated their significant role in optimizing MT. Xu et al.'s research [33] also highlights the importance of these features in optimizing energy consumption related objectives. An interesting observation is that WKR (i.e., the remaining work of the job) and NOR (i.e., the remaining number of jobs) are functionally correlated, both related to the remaining work of jobs. The results reveal a competitive relationship between these two features: when the weight/frequency of WKR is high, the weight/frequency of NOR is low, and vice versa, though they can also both be high simultaneously. This implies inherent correlations among different features, emphasizing the necessity of introducing the feature weight measures in this study.

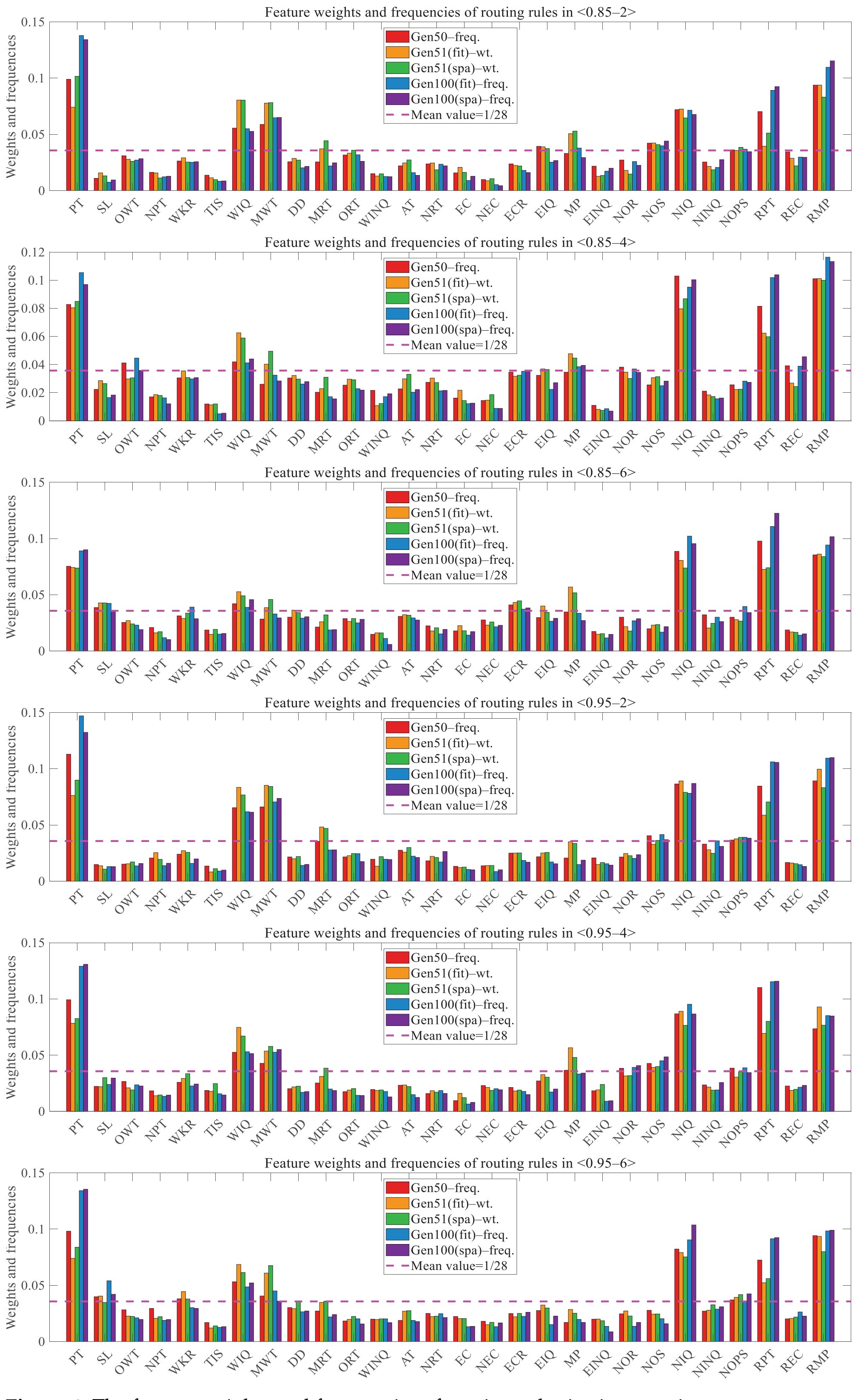

Figure 6. The feature weights and frequencies of routing rules in six scenarios.

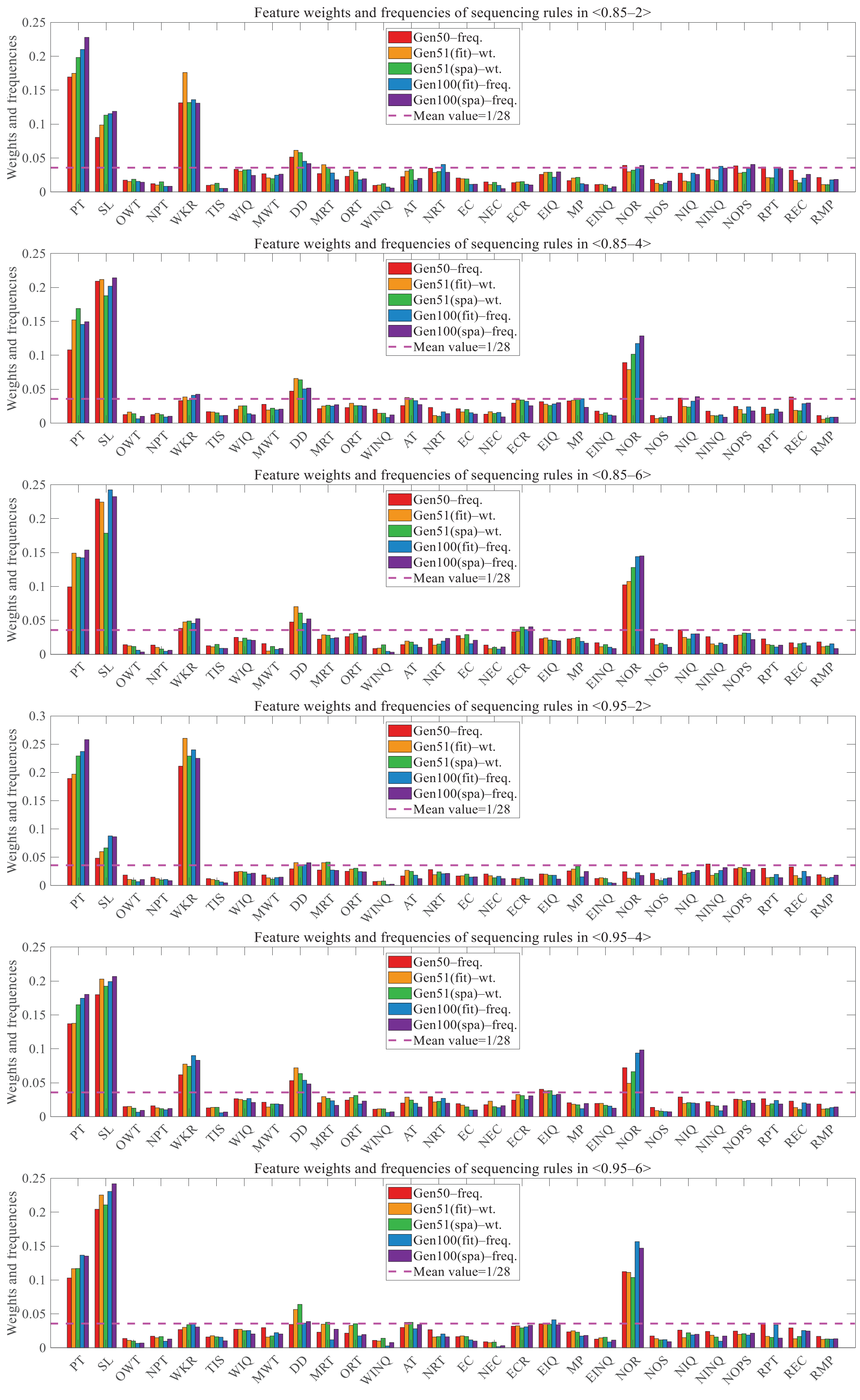

Figure 7. The feature weights and frequencies of sequencing rules in six scenarios.

7. Comprehensive Discussion of the Experimental Results

As numerous experimental results have been presented in Sections 5 and 6, this section offers a comprehensive discussion of the results.

First of all, the experimental results in Sections 5.1 and 5.2 point out that our improved GPHH algorithms (i.e., GPFW(fit) and GPFW(spa)) outperform existing methods (i.e., GPLWT, GPDR, and GPFS) with slightly longer training and testing times. This indicates that using feature weights to guide the evolutionary process of GPHH does benefit the optimization performance when compared to GPDR and surpasses the feature selection method proposed in existing work [31], which focuses more on single-objective problems. Additionally, if we take the experimental results for the number of unique features (in Section 6.1) and the rule size (in Section 6.2) into consideration, one can see that our algorithms not only perform better but also produce more interpretable DRs. This is because the improved GPHH generates DRs with fewer unique features and smaller rule sizes after eliminating irrelevant and redundant features, making the DRs easier to understand.

Considering that the occurrence frequency of each feature in a rule can reflect to some extent the importance or contribution of the feature to the optimization objective, a conjoint analysis of feature weights and feature frequencies is also given in Section 6.3. From the experimental results, one can see that feature frequencies and computed feature weights mostly align. This is because features useful for the optimization objective have larger contributions in individuals/DRs and also occur more frequently. However, feature frequencies can be influenced by irrelevant and redundant features in individuals, limiting the rationality of the results.

In summary, all the above experiments demonstrate that the proposed feature weight measures can more effectively and rationally calculate the significance or contribution of features in the multi-objective DFJSSP compared to existing methods. Hence, they can effectively improve the optimization performance of GPHH and the interpretation of generated DRs by removing irrelevant and redundant features and increasing the occurrence frequencies of high-weight features. Further experiments also point out that the proposed feature weights, in contrast to feature frequencies, better capture the importance and contribution of each feature to the optimization objectives. This is because the feature weights take into account the validity of both individuals and features for the Pareto front.

8. Conclusions and Future Work

This paper addresses the negative impact of irrelevant features on scheduling decisions made by GPHH when generating scheduling rules, as well as the different demands of routing and sequencing scheduling decisions on different feature information. An improved GPHH with dual feature weight sets is proposed and applied to the energy-efficient multi-objective DFJSSP so that high-quality and understandable DRs can be automatically developed.

Specifically, this paper investigates the importance/contribution of different features in solving multi-objective problems and evolving high-quality routing and sequencing rules with GPHH. Then, an improved GPHH with dual feature weight sets is proposed. During the iteration of the population, the importance/contribution of each feature is evaluated based on the fitness of individuals or the sparsity of the Pareto front. In this way, different weights are assigned to the features, which form two different feature weight sets for both routing and sequencing decisions. After this, a hybrid population adjustment strategy is also proposed to genetically update the population based on these newly obtained dual feature weight sets. Experimental results indicate that the proposed improved GPHH with dual feature weight sets surpasses existing related GPHH algorithms and their feature selection methods. The DRs generated by our proposed GPFWs can achieve better optimization performance and interpretability when dealing with multi-objective problems.

In future work, it will be worthwhile to explore the impact of mutation probability on the optimization performance of this proposed GPFW, extend the proposed feature weight measures to many-objective problems, and further improve the population adjustment strategy. These directions could provide valuable exploration for enhancing the algorithm's performance and broader applications.

Author Contributions: Conceptualization, B.X. and B.F.; methodology, K.X. and B.F.; software, B.F.; validation, B.X., K.X., and B.F.; data curation, D.H.; writing—original draft preparation, K.X. and B.F.; writing—review and editing, D.H. and L.T.; supervision, B.X. and Y.W.; funding acquisition, B.X., L.T., and Y.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation (NNSF) of China (Grant No. 62203011), the Young and Middle-aged Teachers Training Action Project of Anhui Province (Grant No. JNFX2023017), the Science and Technology Project of Wuhu (Grant No. 2023jc05), the Scientific Research Foundation for Introduced Talent Scholars, Anhui Polytechnic University (Grant No. 2020YQQ041), the Pre-research Project of the National Natural Science Foundation (NNSF), Anhui Polytechnic University (Grant Nos. Xjky2022042 and Xjky2022048), the Open Fund of Anhui Automotive Display Integrated System Engineering Research Center (Grant No. VDIS2023B02), the Open Fund Key Project of Anhui Provincial Key Laboratory of Detection Technology and Energy Saving Device (Grant No. JCKJ2022A09).

Data Availability Statement: Data are contained within the article.

Acknowledgments: The authors would like to thank the anonymous reviewers and the editor for their positive comments.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

GPHH	Genetic Programming Hyper-Heuristic
DR	Dispatching Rule
DFJSSP	Dynamic Flexible Job Shop Scheduling Problem
MT	Mean Tardiness
TEC	Total Energy Consumption

References

1. Ranade, A.; Gómez, J.; De Juan, A.; Chicaiza, W.D.; Ahern, M.; Escaño, J.M.; Hryshchenko, A.; Casey, O.; Cloonan, A.; O'Sullivan, D.; et al. Implementing Industry 4.0: An In-Depth Case Study Integrating Digitalisation and Modelling for Decision Support System Applications. *Energies* **2024**, *17*, 1818. [CrossRef]
2. Andrei, M.; Thollander, P.; Sannö, A. Knowledge demands for energy management in manufacturing industry-A systematic literature review. *Renew. Sustain. Energy Rev.* **2022**, *159*, 112168. [CrossRef]
3. Zhou, X.; Rao, W.; Liu, Y.; Sun, S. A Decentralized Optimization Algorithm for Multi-Agent Job Shop Scheduling with Private Information. *Mathematics* **2024**, *12*, 971. [CrossRef]
4. Jiang, B.; Ma, Y.; Chen, L.; Huang, B.; Huang, Y.; Guan, L. A Review on Intelligent Scheduling and Optimization for Flexible Job Shop. *Int. J. Control Autom. Syst.* **2023**, *21*, 3127–3150. [CrossRef]
5. Li, X.; Guo, X.; Tang, H.; Wu, R.; Wang, L.; Pang, S.; Liu, Z.; Xu, W.; Li, X. Survey of integrated flexible job shop scheduling problems. *Comput. Ind. Eng.* **2022**, *174*, 108786. [CrossRef]
6. Destouet, C.; Tlahig, H.; Bettayeb, B.; Mazari, B. Flexible job shop scheduling problem under Industry 5.0: A survey on human reintegration, environmental consideration and resilience improvement. *J. Manuf. Syst.* **2023**, *67*, 155–173. [CrossRef]
7. Zhang, F.; Mei, Y.; Nguyen, S.; Zhang, M. Survey on Genetic Programming and Machine Learning Techniques for Heuristic Design in Job Shop Scheduling. *IEEE Trans. Evol. Comput.* **2024**, *28*, 147–167. [CrossRef]
8. Lunardi, W.T.; Birgin, E.G.; Laborie, P.; Ronconi, D.P.; Voos, H. Mixed integer linear programming and constraint programming models for the online printing shop scheduling problem. *Comput. Oper. Res.* **2020**, *123*, 105020. [CrossRef]
9. Gmys, J.; Mezmaz, M.; Melab, N.; Tuyttens, D. A computationally efficient Branch-and-Bound algorithm for the permutation flow-shop scheduling problem. *Eur. J. Oper. Res.* **2020**, *284*, 814–833. [CrossRef]
10. Lin, S.W.; Ying, K.C. Minimising makespan in job-shops with deterministic machine availability constraints. *Int. J. Prod. Res.* **2021**, *59*, 4403–4415. [CrossRef]
11. Firme, B.; Figueiredo, J.; Sousa, J.M.; Vieira, S.M. Agent-based hybrid tabu-search heuristic for dynamic scheduling. *Eng. Appl. Artif. Intell.* **2023**, *126*, 107146. [CrossRef]
12. Fontes, D.B.; Homayouni, S.M.; Gonçalves, J.F. A hybrid particle swarm optimization and simulated annealing algorithm for the job shop scheduling problem with transport resources. *Eur. J. Oper. Res.* **2023**, *306*, 1140–1157. [CrossRef]
13. Sun, M.; Cai, Z.; Zhang, H. A teaching-learning-based optimization with feedback for LR fuzzy flexible assembly job shop scheduling problem with batch splitting. *Expert Syst. Appl.* **2023**, *224*, 120043. [CrossRef]

14. Cheng, L.; Tang, Q.; Zhang, L. Mathematical model and adaptive simulated annealing algorithm for mixed-model assembly job-shop scheduling with lot streaming. *J. Manuf. Syst.* **2023**, *70*, 484–500. [CrossRef]
15. Zhu, K.; Gong, G.; Peng, N.; Zhang, L.; Huang, D.; Luo, Q.; Li, X. Dynamic distributed flexible job-shop scheduling problem considering operation inspection. *Expert Syst. Appl.* **2023**, *224*, 119840. [CrossRef]
16. Zhang, B.; Pan, Q.K.; Meng, L.L.; Zhang, X.L.; Jiang, X.C. A decomposition-based multi-objective evolutionary algorithm for hybrid flowshop rescheduling problem with consistent sublots. *Int. J. Prod. Res.* **2023**, *61*, 1013–1038. [CrossRef]
17. Zhang, H.; Buchmeister, B.; Li, X.; Ojstersek, R. An Efficient Metaheuristic Algorithm for Job Shop Scheduling in a Dynamic Environment. *Mathematics* **2023**, *11*, 2336. [CrossRef]
18. Zhang, F.; Mei, Y.; Nguyen, S.; Zhang, M. Collaborative multifidelity-based surrogate models for genetic programming in dynamic flexible job shop scheduling. *IEEE Trans. Cybern.* **2021**, *52*, 8142–8156. [CrossRef]
19. Wang, H.; Peng, T.; Nassehi, A.; Tang, R. A data-driven simulation-optimization framework for generating priority dispatching rules in dynamic job shop scheduling with uncertainties. *J. Manuf. Syst.* **2023**, *70*, 288–308. [CrossRef]
20. Xiong, H.; Wang, H.; Shi, S.; Chen, K. Comparison study of dispatching rules and heuristics for online scheduling of single machine scheduling problem with predicted release time jobs. *Expert Syst. Appl.* **2024**, *243*, 122752. [CrossRef]
21. Panwalkar, S.S.; Iskander, W. A survey of scheduling rules. *Oper. Res.* **1977**, *25*, 45–61. [CrossRef]
22. Zhang, F.; Mei, Y.; Nguyen, S.; Zhang, M. Correlation coefficient-based recombinative guidance for genetic programming hyperheuristics in dynamic flexible job shop scheduling. *IEEE Trans. Evol. Comput.* **2021**, *25*, 552–566. [CrossRef]
23. Zhou, Y.; Yang, J.J.; Huang, Z. Automatic design of scheduling policies for dynamic flexible job shop scheduling via surrogate-assisted cooperative co-evolution genetic programming. *Int. J. Prod. Res.* **2020**, *58*, 2561–2580. [CrossRef]
24. Fan, H.; Xiong, H.; Goh, M. Genetic programming-based hyper-heuristic approach for solving dynamic job shop scheduling problem with extended technical precedence constraints. *Comput. Oper. Res.* **2021**, *134*, 105401. [CrossRef]
25. Rosca, J.P.; Ballard, D.H. *Genetic Programming with Adaptive Representations*; Department of Computer Science, University of Rochester: Rochester, NY, USA, 1994.
26. Mei, Y.; Nguyen, S.; Xue, B.; Zhang, M. An efficient feature selection algorithm for evolving job shop scheduling rules with genetic programming. *IEEE Trans. Emerg. Top. Comput. Intell.* **2017**, *1*, 339–353. [CrossRef]
27. Sitahong, A.; Yuan, Y.; Li, M.; Ma, J.; Ba, Z.; Lu, Y. Learning dispatching rules via novel genetic programming with feature selection in energy-aware dynamic job-shop scheduling. *Sci. Rep.* **2023**, *13*, 8558. [CrossRef] [PubMed]
28. Friedlander, A.; Neshatian, K.; Zhang, M. Meta-learning and feature ranking using genetic programming for classification: Variable terminal weighting. In Proceedings of the 2011 IEEE Congress of Evolutionary Computation (CEC), New Orleans, LA, USA, 5–8 June 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 941–948.
29. Mei, Y.; Zhang, M.; Nyugen, S. Feature selection in evolving job shop dispatching rules with genetic programming. In Proceedings of the Genetic and Evolutionary Computation Conference, Denver, CO, USA, 20–24 July 2016; pp. 365–372.
30. Zhang, F.; Mei, Y.; Zhang, M. A two-stage genetic programming hyper-heuristic approach with feature selection for dynamic flexible job shop scheduling. In Proceedings of the Genetic and Evolutionary Computation Conference, Prague, Czech Republic, 13–17 July 2019; pp. 347–355.
31. Zhang, F.; Mei, Y.; Nguyen, S.; Zhang, M. Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job-shop scheduling. *IEEE Trans. Cybern.* **2020**, *51*, 1797–1811. [CrossRef] [PubMed]
32. Jia, S.; Yang, Y.; Li, S.; Wang, S.; Li, A.; Cai, W.; Liu, Y.; Hao, J.; Hu, L. The Green Flexible Job-Shop Scheduling Problem Considering Cost, Carbon Emissions, and Customer Satisfaction under Time-of-Use Electricity Pricing. *Sustainability* **2024**, *16*, 2443. [CrossRef]
33. Xu, B.; Mei, Y.; Wang, Y.; Ji, Z.; Zhang, M. Genetic programming with delayed routing for multiobjective dynamic flexible job shop scheduling. *Evol. Comput.* **2021**, *29*, 75–105. [CrossRef]
34. Shen, L.; Dauzère-Pérès, S.; Maecker, S. Energy cost efficient scheduling in flexible job-shop manufacturing systems. *Eur. J. Oper. Res.* **2023**, *310*, 992–1016. [CrossRef]
35. Branke, J.; Nguyen, S.; Pickardt, C.W.; Zhang, M. Automated design of production scheduling heuristics: A review. *IEEE Trans. Evol. Comput.* **2015**, *20*, 110–124. [CrossRef]
36. Tian, Y.; Gao, Z.; Zhang, L.; Chen, Y.; Wang, T. A Multi-Objective Optimization Method for Flexible Job Shop Scheduling Considering Cutting-Tool Degradation with Energy-Saving Measures. *Mathematics* **2023**, *11*, 324. [CrossRef]
37. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
38. Luan, F.; Cai, Z.; Wu, S.; Liu, S.Q.; He, Y. Optimizing the low-carbon flexible job shop scheduling problem with discrete whale optimization algorithm. *Mathematics* **2019**, *7*, 688. [CrossRef]
39. Zhang, F.; Mei, Y.; Zhang, M. Genetic programming with multi-tree representation for dynamic flexible job shop scheduling. In Proceedings of the AI 2018: Advances in Artificial Intelligence: 31st Australasian Joint Conference, Proceedings 31, Wellington, New Zealand, 11–14 December 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 472–484.
40. Tay, J.C.; Ho, N.B. Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. *Comput. Ind. Eng.* **2008**, *54*, 453–473. [CrossRef]
41. Nguyen, S.; Zhang, M.; Johnston, M.; Tan, K.C. Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming. *IEEE Trans. Evol. Comput.* **2013**, *18*, 193–208. [CrossRef]

42. Zeiträg, Y.; Figueira, J.R.; Horta, N.; Neves, R. Surrogate-assisted automatic evolving of dispatching rules for multi-objective dynamic job shop scheduling using genetic programming. *Expert Syst. Appl.* **2022**, *209*, 118194. [CrossRef]
43. Ishibuchi, H.; Imada, R.; Setoguchi, Y.; Nojima, Y. How to specify a reference point in hypervolume calculation for fair performance comparison. *Evol. Comput.* **2018**, *26*, 411–440. [CrossRef]
44. Wilcoxon, F. Individual comparisons by ranking methods. In *Breakthroughs in Statistics: Methodology and Distribution*; Springer: Berlin/Heidelberg, Germany, 1992; pp. 196–202.
45. Shady, S.; Kaihara, T.; Fujii, N.; Kokuryo, D. A novel feature selection for evolving compact dispatching rules using genetic programming for dynamic job shop scheduling. *Int. J. Prod. Res.* **2022**, *60*, 4025–4048. [CrossRef]