



Research article

Surrogate-assisted automatic design of dispatching rules using ensemble genetic programming for multi-objective dynamic job shop scheduling

Gang Wang¹, Adilanmu Sitahong^{1,*}, Yiping Yuan¹, Peiyin Mo¹, Areziguli Wubuli² and Junyan Ma²

¹ School of Mechanical Engineering, Xinjiang University, Urumqi 830047, China

² School of Economic and Management, Xinjiang University, Urumqi 830047, China

* **Correspondence:** Email: adilanjmu@xju.edu.cn.

Abstract: In multi-objective dynamic job shop scheduling (MODJSS), achieving production efficiency while reducing energy consumption remains a major challenge in green manufacturing. Genetic programming (GP) has been widely adopted as an effective hyper-heuristic approach for automatically evolving dispatching rules (DRs). Surrogate models have also been widely used in GP frameworks to reduce computational cost. However, existing surrogate-assisted approaches mainly focus on a single DR and cannot capture the collective behavior of ensemble individuals, that is, the interactions and possible conflicts among multiple rules. To address this issue, this paper proposes an improved surrogate-assisted evaluation framework based on ensemble genetic programming (EGP) to capture the collective behavior of ensemble individuals and thereby reduce the computational cost of simulation-based fitness evaluation. Phenotype-based feature representations were developed for both individual and ensemble rules, where ensemble features incorporate behavioral information, diversity, and decision conflict. Surrogate models were trained to predict multiple objectives, and a hierarchical evaluation strategy was employed in which candidate solutions are first evaluated by surrogate predictions and ranked by non-dominated sorting, while only a subset of high-quality solutions is selected for actual simulation. The experimental results showed that (i) the proposed framework significantly reduces the number of expensive simulation evaluations and the evaluation cost of ensemble individuals, while remaining competitive in solution quality; (ii) surrogate-assisted evaluation improves the computational efficiency of EGP and indicates that the method has application potential in larger and more complex dynamic scheduling scenarios.

Keywords: genetic programming (GP); multi-objective job shop scheduling; surrogate-assisted evaluation; ensemble; dispatching rules

Mathematics Subject Classification: 90B35, 68W50

1. Introduction

Against the backdrop of continuously growing global energy demand and increasingly severe climate change, energy consumption and carbon emissions have become key factors constraining sustainable economic development [1]. According to relevant statistics, the industrial sector, as one of the major sources of energy consumption, accounts for a significant proportion of global energy use, with manufacturing being a particularly energy-intensive component [2]. Therefore, under these circumstances, the manufacturing industry must seek effective strategies to reduce energy consumption and carbon emissions. Currently, enterprises primarily focus on optimization at two levels: the machine level and the production level [3]. However, upgrading machinery often entails long research and development cycles and substantial investment costs, creating a formidable barrier for many companies [4]. Hence, this paper focuses on the production level and seeks to reduce energy consumption through job shop scheduling, without upgrading machines.

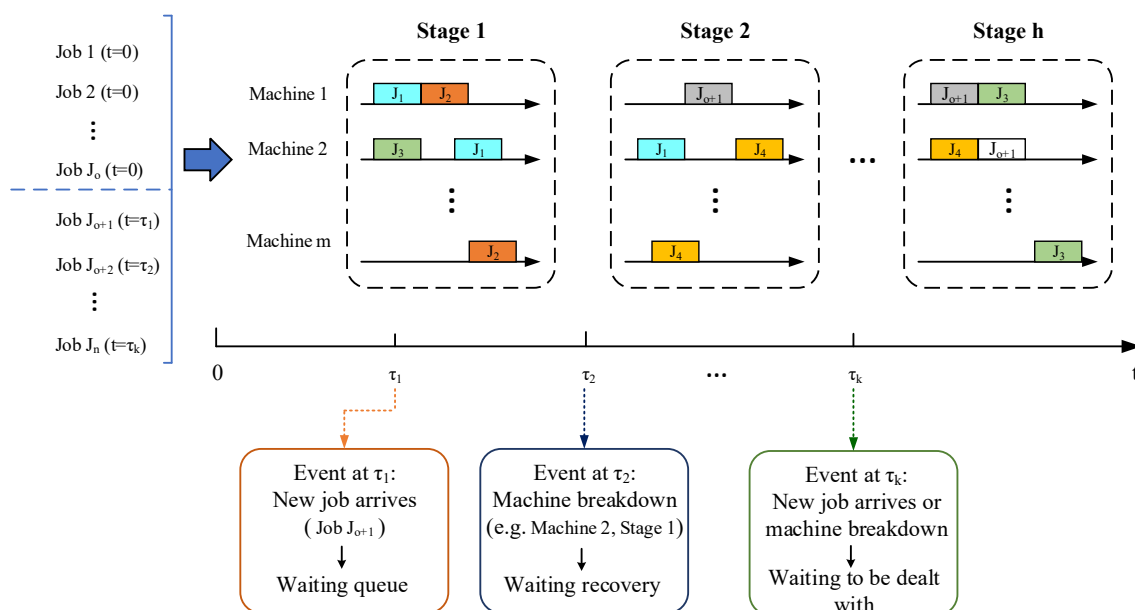


Figure 1. The layout of the DJSS.

As one of the core decision-making problems in manufacturing, job shop scheduling (JSS) directly affects equipment operating status, processing sequences, and energy utilization efficiency through its scheduling strategies [5]. In real manufacturing environments, however, factors such as dynamic job arrivals, changes in equipment status, and uncertainties in processing [6,7] render scheduling problems highly dynamic and complex [8]. These dynamic disturbances must be considered in order to represent real workshop processing scenarios. In addition, manufacturing

companies often need to strive to achieve one goal while also balancing several other potentially conflicting goals, such as delivery time and production costs [9,10]. Therefore, multi-objective dynamic job shop scheduling (MODJSS), which is closely related to real-world processing scenarios, is the focus of this paper. Figure 1 shows the layout of the DJSS.

As a heuristic approach, dispatching rules (DRs) exhibit low time complexity and can respond quickly to uncertain events in actual production, making them very popular in practical scheduling applications [11]. However, designing high-performance dispatching rules for specific production objectives is a highly challenging task. Genetic programming hyper-heuristics (GPHH), as a machine learning technique, have been widely applied in job shop scheduling due to their ability to automatically construct scheduling rule expressions [12–14]. However, the scheduling performance of GPHH is highly dependent on the quality of the feature set design. On the one hand, if the features fail to adequately reflect the dynamic state of the workshop, the generated rules will struggle to learn effective scheduling behaviors [15]; on the other hand, as the number of feature dimensions increases, the rule expression space expands rapidly, leading to increased search complexity and reduced evolutionary stability [16].

In an effort to further improve scheduling performance and enhance solution robustness, researchers have introduced ensemble learning into genetic programming, resulting in ensemble genetic programming (EGP) [17]. This method combines multiple dispatching rules, allowing different rules to leverage their respective advantages under various decision scenarios, thereby improving overall scheduling effectiveness [18]. However, while the ensemble mechanism brings performance gains, it also significantly increases computational complexity. At each scheduling decision point, an ensemble individual must invoke multiple member rules for evaluation and voting [19], making the cost of a single simulation evaluation much higher than that of a single individual. Moreover, as the ensemble size grows, the interactions among rules become more complex, further amplifying the time consumption of the evaluation process. This issue is particularly pronounced in dynamic scheduling scenarios, where traditional simulation-based evaluation struggles to meet real-time and efficiency requirements.

One promising approach to reducing the cost of simulation evaluation is the use of surrogate models, which have been widely adopted in evolutionary optimization to approximate the fitness of candidate solutions via machine learning methods, thereby reducing the number of expensive real evaluations [20]. In job shop scheduling, existing studies typically construct feature representations of dispatching rules and use regression models to predict their performance under different objectives [21]. However, most current surrogate-assisted methods are designed for individual dispatching rules, with feature representations often based on the behavioral responses of a rule in several typical scenarios, making them difficult to extend directly to ensemble individuals [12]. Compared with a single rule, the scheduling behavior of an ensemble individual is jointly determined by multiple rules, and its decision outcomes depend not only on the individual performance of each member rule but also on the synergies and conflicts among them. Therefore, simply applying a linear combination of individual features cannot accurately characterize the overall behavior of an ensemble individual. Furthermore, existing approaches typically embed the surrogate model as an auxiliary evaluation tool within the evolutionary process but lack a unified evaluation mechanism for hybrid populations containing both individuals and ensemble individuals, thereby limiting the effectiveness of surrogate models in complex evolutionary frameworks.

In response to the aforementioned challenges, a surrogate-assisted evaluation framework for EGP is developed in this work. First, based on the behavioral responses of dispatching rules in typical

decision scenarios, a unified phenotypic feature representation method is constructed to map both single individuals and ensemble individuals into a common feature space. For ensemble individuals, diversity measures and decision conflict information are further introduced to characterize their internal structural properties and behavioral differences. Second, multi-objective surrogate models are built on this feature space, enabling efficient performance prediction of candidate solutions. Furthermore, a dynamic hierarchical evaluation mechanism is designed: candidate solutions are first predicted by the surrogate model and then ranked through non-dominated sorting based on the predicted results. Only a subset of solutions with superior rankings undergo real simulation evaluation, while the remaining solutions retain their predicted values for participation in the evolutionary process. The real-evaluation budget is also adjusted during evolution. Through these mechanisms, this paper aims to balance solution quality and real simulation cost. The main contributions of this paper are as follows:

(1) A unified phenotypic feature representation method is proposed, which maps single dispatching rules and ensemble rules into the same behavioral feature space. Structural information is further introduced for ensemble individuals, which provides a basis for unified surrogate modeling of the hybrid individual–ensemble population.

(2) Surrogate models are used to predict candidate individuals quickly, and a subset of candidates is selected for real simulation according to the predicted performance, which improves the use of a limited real-evaluation budget.

(3) To address the difficulty in characterizing ensemble individuals, a structure-aware ensemble feature representation mechanism is constructed. On the basis of behavioral response features, it further fuses the diversity and decision conflict information among rules, so as to improve the ability of the surrogate model to distinguish performance differences among ensemble policies.

The remainder of this paper is structured as follows: Section 2 reviews relevant research on GP, EGP, and surrogate models in MODJSS. Section 3 introduces the notations and mathematical model of DJSS. Section 4 provides a detailed description of the proposed algorithm's framework and the specifics of each module. Section 5 details the experimental environment and parameter settings. Section 6 presents the experimental analysis, including performance comparisons and the accuracy of the surrogate model. Section 7 concludes the paper.

2. Literature review

2.1. Energy-saving optimization job shop scheduling

In recent years, as the demand for energy saving and emission reduction has increased, incorporating energy consumption into the optimization objectives has become an important direction in manufacturing systems research. Traditional job shop scheduling mainly focuses on production-efficiency indicators such as makespan and maximum tardiness, whereas green job shop scheduling further considers the energy consumed during processing, so that dispatching decisions can balance production efficiency and energy use. Existing solution methods for job shop scheduling can be broadly divided into exact methods and approximate methods [22]. Because the problem is NP-hard, exact methods are difficult to apply to medium- and large-scale instances. Research has therefore focused on approximate methods, including heuristics [23], metaheuristics [24], and learning-based intelligent optimization methods [25].

In energy-efficient job shop scheduling, energy consumption is closely related to the processing, idle, and start-up/shut-down states of machines. Existing studies therefore reduce energy consumption at mainly two levels: equipment operation control and production scheduling decisions. One line of work reduces non-processing energy by controlling machine operating states. For example, Tian et al. [26] optimized energy saving by controlling tool replacement and machine start/stop, using a hybrid tool-machine energy-saving strategy to save a substantial portion of non-processing energy. Although researchers have demonstrated that controlling machine start/stop can effectively save energy, frequent switching on and off may cause irreversible damage to machines [27], and in large-scale factories such as those in the automotive and steel industries, shutting down machines during production is generally not permitted except under special circumstances [28].

Another line of work reduces energy consumption by adjusting the temporal distribution of processing tasks, among which time-of-use electricity pricing and machine speed adjustment are typical strategies. Time-of-use pricing shifts production tasks toward off-peak periods and thereby reduces energy cost. For instance, Xu et al. [29] established a mathematical model considering both time-of-use pricing and tiered electricity pricing to reduce electrical energy consumption during production and assembly. However, time-of-use pricing only reflects the patterns of power load and supply across different periods. Due to cost constraints, grid operators cannot update prices in real time, which inevitably increases the operational risks for enterprises [30].

By contrast, machine speed adjustment reduces energy consumption by controlling machine processing speeds. Reducing machine speed generally lowers processing energy consumption; that is, slowing down certain non-critical operations can save energy without affecting overall production efficiency [31]. For example, Shao et al. [32] proposed a multi-neighborhood-based multi-objective metaheuristic algorithm (MMMA) to optimize total weighted tardiness and energy consumption. The MMMA adopts a two-layer encoding scheme consisting of job permutation and speed matrix to solve the distributed flexible flow shop scheduling problem, demonstrating that the machine speed adjustment strategy significantly contributes to energy saving.

In addition to the strategies above, researchers have gradually turned to energy-efficient scheduling in dynamic production environments as manufacturing systems become more complex. Unlike a static job shop, a dynamic job shop is subject to disturbances such as random job arrivals and machine breakdowns, which may invalidate an existing schedule in a short time. Energy saving, therefore, cannot be obtained by static optimization alone; dispatching decisions must be adjusted as the shop state changes. For dynamic energy-efficient scheduling, some studies improve the algorithm so as to reduce energy loss under dynamic shop conditions. For example, Duan et al. [33] considered the requirements of dynamic energy-efficient scheduling in workshops. They designed two terminal sets related to dynamic events and energy saving and assigned weight coefficients to each terminal according to workshop states when generating dispatching rules, so as to enhance rule performance. Hou et al. [34] addressed the dynamic energy-efficient flexible job shop scheduling problem by proposing an improved multi-objective genetic programming (IMOGP) method based on Pareto dominance. This method uses terminal symbols specifically designed for machine breakdowns and designs an adaptive mutation strategy. In this strategy, the frequency of occurrence of terminals in the high-quality Pareto population is abstracted as their importance, and the mutation probability of terminals is adjusted according to disturbance factors. Sitahong et al. [35] proposed a genetic programming method with online feature selection to automatically learn dispatching rules. Promising individuals evolved by GP are used to select key feature sets, and then, more competitive rules are retrained.

To sum up, job shop scheduling under green manufacturing has evolved from single time-efficiency optimization to multi-objective collaborative optimization and has further considered practical factors such as machine start/stop, time-of-use pricing, processing-speed adjustment, and stochastic dynamic events. However, existing energy-saving strategies usually rely on specific equipment-control schemes, energy-price mechanisms, or manually designed mathematical models. For a dynamic shop that simultaneously involves random job arrivals, machine breakdowns, and multiple objectives, the decision space is more complex, and fixed dispatching policies can hardly adapt to the changing shop state. Therefore, how to construct dispatching rules with stronger adaptability and generalization from the shop state automatically has become an important direction for further research on dynamic energy-efficient job shop scheduling.

2.2. Ensemble genetic programming

Compared with manually designed dispatching rules, the main advantage of GPHH is that it can automatically learn dispatching logic in the rule space, rather than searching for a schedule for one specific instance. When the shop state changes, the trained rules can therefore produce dispatching decisions from the current state. This pattern of offline rule learning and online rapid decision-making makes GPHH particularly suitable for DJSS. A single dispatching rule, however, still has limitations. Different rules usually have advantages in different shop states: a rule may perform well in some decision situations and poorly in others. When random job arrivals and machine breakdowns occur together, this situation dependence may become stronger. Relying on one evolved rule alone may therefore fail to exploit complementary information among different rules.

To improve the robustness of dispatching rules, ensemble learning has been introduced into genetic programming, yielding ensemble genetic programming (EGP). As one of the leading methods in machine learning [36], ensemble learning combines multiple baseline models [37]. Its core purpose is to construct a more powerful composite model than its individual components and thereby reduce the risk of overfitting. In recent years, research on the combination of ensemble learning and genetic programming has been discussed by many researchers [38]. Hart and Sim [39] first proposed a novel hyper-heuristic method, NELLI-GP, to solve JSS. This method extends the existing ensemble method NELLI to evolve a set of heuristics. The set adopts a divide-and-conquer strategy, where each heuristic corresponds to a unique subset of the instance set. Genetic programming under the ensemble approach improves performance by combining multiple rules, but most of these approaches adopt majority voting. To address this, Park et al. [40] investigated four combination schemes for ensemble rules—majority voting, linear combination, weighted majority voting, and weighted linear combination—and improved the coevolutionary framework to evolve both rules and weights simultaneously. Park et al. [41] applied cooperative coevolutionary GP (CCGP) to static job shop scheduling. This method comprises as many subpopulations as there are members in the ensemble, and each subpopulation evolves a single scheduling heuristic for the ensemble.

Although many researchers have confirmed the advantages of combining ensemble learning with genetic programming, the shops addressed in these studies are predominantly static. Xu et al. [18] proposed a novel EGP method for dynamic shops, in which single individuals and ensemble individuals jointly participate in evolution. The experimental results demonstrate that the designed method can effectively enhance the performance of EGP.

In summary, ensemble mechanisms mitigate the sharp performance degradation of a single dispatching rule under dynamic disturbances through complementarity among multiple models. This gain, however, comes at the cost of computational efficiency: for a single DR, one simulation invokes only one rule at each decision point, whereas an ensemble individual must invoke multiple member rules and then combine or vote. As the ensemble size increases, the simulation cost of one ensemble individual is substantially higher than that of an ordinary individual. Particularly in complex dynamic instances involving multi-objective optimization, intensive fitness evaluation has become a major bottleneck restricting the application of EGP. Therefore, how to reduce the expensive simulation cost while retaining the performance advantage of EGP remains a key issue for further research.

2.3. Surrogate-assisted GPHH

Prior to this, most researchers focused on improving GPHH so that the automatically generated dispatching rules would perform better. For example, Huang et al. [42] proposed a multi-task linear genetic programming (LGP) method that transfers knowledge through multi-output individuals, that is, individuals shared across tasks, to improve population performance.

In GPHH, the fitness of a candidate dispatching rule is usually obtained by discrete-event simulation. This evaluation can reflect the actual scheduling performance of a rule in the target shop, but the evolutionary process contains a large number of candidate rules and, therefore, requires many repeated simulations, which leads to a high computational cost. This cost is especially high in MODJSS, where random job arrivals and machine breakdowns mean that a single rule usually has to be evaluated in multiple dynamic scenarios; for EGP, each ensemble individual further contains multiple member rules, which increases the cost of one simulation. Introducing a surrogate model to reduce expensive real simulations has therefore become an important direction for improving the computational efficiency of GPHH.

At present, surrogate-assisted evaluation methods fall into two types. The first type improves evaluation efficiency by appropriately simplifying the simulation scenarios. For example, Mei et al. [43] developed a GPHH framework based on niche search to extract DRs and employed a surrogate model to evaluate the fitness of DRs to reduce the complexity of fitness evaluation. Experimental verification demonstrated that this method reduced the evaluation time by 10%. Zhou et al. [44] proposed a surrogate-assisted coevolutionary genetic programming method (CCGP-SM). This method introduces the HalfShop simplified simulation model proposed by Nguyen et al. [45] as a surrogate, and accelerates evaluation by reducing the number of machines and jobs. In evolution, the surrogate model is used to screen the intermediate population quickly, and only the best individuals of each generation receive high-accuracy real evaluation. Zhang et al. [15] proposed a genetic programming method (M³GP²) based on collaborative multi-fidelity surrogate models. In this method, several simplified simulation models with different fidelities are designed as surrogates, the population is divided into a corresponding number of subpopulations for parallel evolution, and knowledge is transferred among subpopulations through a subtree-based crossover mechanism.

The second type of surrogate-assisted method does not rely on a simplified model but estimates fitness through similarity among individuals. For example, Hildebrandt and Branke [46] introduced individual phenotypic characterization to encode the behavior of dispatching rules in typical decision situations as decision vectors and then used k-nearest neighbor (KNN) regression to predict fitness. The core idea of this method is that rules with similar behavior often have similar performance, so the

fitness of new individuals can be estimated from historical samples, and a full simulation can be avoided. However, the prediction accuracy of KNN is highly dependent on the construction of the feature space and on sample coverage. How to design more discriminative phenotypic features is still a problem worthy of further study. On this basis, Nguyen et al. [47] proposed a people-centric evolutionary system (PES), which combines phenotypic features with topology learning and uses improved growing neural gas (mGNG) to model historical search data incrementally and capture the distribution of heuristic rules in the search space. Unlike static KNN prediction, the adaptive surrogate constructed by PES introduces three dynamic factors—diversity control, expansion control, and user preference—and adjusts the estimated fitness according to the neighborhood density of the new individual in the topology network, the program size, and the proximity to a user-selected region, thereby guiding the search while making predictions.

However, PES and the KNN surrogate on which it depends mainly use the current generation as the sample library. In dynamic scheduling, training instances rotate with generations, so the fitness values of individuals from different generations are not comparable, and historical data cannot be used effectively. In view of this limitation, Zhang et al. [48] proposed a surrogate model based on instance rotation. Its core innovation is a fitness-mapping strategy: the common individuals retained by elite selection and reproduction between two adjacent generations are used to learn a linear mapping from the previous generation to the current generation, and historical fitness values are converted into the current comparable space. The surrogate can therefore fuse samples from multiple generations and substantially enlarge the KNN reference library. Experiments show that this strategy improves the accuracy of fitness estimation and enables genetic programming to screen promising individuals more accurately from large offspring sets and to learn better dispatching rules.

Although existing surrogate-assisted methods have made substantial progress in improving the evolutionary efficiency of conventional GPHH, most of them focus on a single DR. In the context of EGP, dispatching decisions are made jointly by single individuals and ensemble individuals. Existing surrogate frameworks therefore face a sharp increase in the dimensionality and complexity of phenotypic features, and the tension between evaluation cost and prediction accuracy becomes more severe. Furthermore, surrogate predictions inevitably contain errors. In multi-objective evolution, such errors may not only affect the fitness estimate of a single individual but also change the non-dominated relations and Pareto ranking among candidates. If a large number of candidates rely entirely on surrogate predictions, the errors may accumulate across generations and affect the search direction. For EGP, it is therefore not sufficient to build a surrogate that can predict ensemble performance; a coordinated mechanism of surrogate prediction, real-simulation screening, and error control is also needed, so that low-cost prediction and high-accuracy real evaluation can work together.

In summary, existing surrogate-assisted studies have made good progress in reducing the simulation cost of GPHH, but three main gaps remain. First, existing methods mainly target a single DR and pay insufficient attention to ensemble individuals composed of multiple cooperating DRs. Second, existing phenotypic features mainly describe the behavior of a single rule and can hardly characterize, at the same time, the behavioral differences, diversity, and decision conflict among ensemble members. Third, existing surrogate-assisted evaluation mechanisms have not fully solved unified evaluation in an individual–ensemble hybrid population or the control of surrogate prediction error.

Therefore, how to provide sufficient prediction accuracy while minimizing the loss of ensemble gain is the problem that this paper aims to solve. At present, there is no mature surrogate mechanism that can provide accurate and low-cost performance estimation for ensemble individuals, which have

a more complex structure and a higher computational cost. In response to this research gap, this paper proposes a surrogate-assisted evaluation framework designed specifically for EGP, in order to improve its computational efficiency and performance.

3. Problem description

3.1. Assumptions and symbolic descriptions

In the JSS, a batch of jobs $J = \{J_1, J_2, \dots, J_n\}$ is to be processed on a set of m machines $M = \{M_1, M_2, \dots, M_m\}$. Unlike the classical JSS, this study considers two dynamic factors: machine breakdowns and random job arrivals. The scheduling objective is to minimize the makespan and the total energy consumption by arranging the processing sequence of jobs.

The makespan (C_{max}), as the most frequently adopted scheduling objective in current job shop scheduling, can most intuitively reflect the efficiency of the current schedule by calculating the completion time of the last operation of the last job. It is also the most intuitive indicator of an enterprise's processing capacity.

Table 1. Notations and variables.

Symbol	Meaning
J	Set of jobs, $J = \{1, 2, \dots, n\}$
M	Set of machines, $M = \{1, 2, \dots, m\}$
i, l	Index of machines
j, k	Index of jobs
h	Index of operations
h_j	Total number of operations of job j
O_{jhi}	The h -th process of job j processed on machine i
C_j	The total completion time of job j
P_{ijh}	The processing time of the h -th operation of job j on machine i
S_{jh}	The start time of the h -th process for job j
C_{jh}	The completion time of the h -th process for job j
L	A positive number
d_j	Due date of job j
r_j	The arrival time of job j
R_i	The maintenance/recovery time after a breakdown of machine i
$x_{jh_{it}}$	1 if operation h of job j is processed on machine i at time t ; otherwise, 0
z_i^t	1 if machine i breaks down at time t ; otherwise, 0

This paper takes total energy consumption as the second objective. The focus is the energy-use efficiency of the manufacturing system at the operational level, rather than a direct thermodynamic

model of irreversibility in processing. In the dynamic job shop studied here, the dispatching decision determines not only the job sequence but also the processing duration, operating speed, idle time, and the recovery process after a breakdown. The same set of jobs may therefore produce substantially different energy consumption under different dispatching rules.

To solve this multi-objective problem, a Pareto optimality-based approach is adopted to find a set of non-dominated solutions that satisfy the two optimization objectives. The symbol descriptions are presented in Table 1. The assumptions of this problem are as follows:

- (1) A job can only be processed on one machine at a time, and one machine can only process one job at a time.
- (2) Each machine is available at time zero, and there is an unlimited buffer between machines.
- (3) This study does not consider operation setup time or transportation time.
- (4) The machine breakdown rate follows an exponential distribution, the job arrival time follows a Poisson distribution, and the machine maintenance time follows a uniform distribution.
- (5) During the simulation process, machine breakdowns are generated randomly and transmitted to the mathematical model in the form of real-time data, which is denoted by z_i^t . For a given machine i , if it fails at time t , then $z_i^t = 1$; otherwise, $z_i^t = 0$.
- (6) When the same operation of the same job is being processed, the machine can encounter at most one breakdown.
- (7) During breakdown maintenance, the machine is charged at idle power.
- (8) Each machine has equal startup and shutdown power, as well as equal startup and shutdown time.

3.2. Mathematical model

The mathematical model for energy-saving scheduling in a dynamic job shop, considering machine breakdowns and random arrival of jobs, is established as follows:

$$\text{Min}(C_{\max}, E_{\text{total}}) \quad (1)$$

$$C_{\max} = \text{Max}_{1 \leq j \leq n}(C_j) \quad (2)$$

$$E_{\text{total}} = \text{REc} + \text{PEc} + \text{IEc} \quad (3)$$

where REc, PEc, and IEc represent ready energy consumption, processing energy consumption, and idle energy consumption, respectively.

(1) Ready energy consumption (REc): Refers to the energy used to start up the equipment before processing and to shut it down after processing, which can be expressed as:

$$\text{REc} = \sum_{i=1}^m P_i^{\text{setup}} t_i^{\text{setup}} \times \alpha_i \quad (4)$$

where P_i^{setup} denotes the start-up/shut-down power of the machine, t_i^{setup} denotes the start-up / shut-down time, and α_i denotes the number of start-up / shut-down operations.

(2) Processing energy consumption (PEc): Refers to the energy consumption during the processing of jobs, which can be expressed as:

$$PEc = \sum_{i=1}^m \sum_{j=1}^n \sum_{h=1}^w P_{iq}^{pro} \times t_{ijh}^{pro} \quad (5)$$

where P_{iq}^{pro} and t_{ijh}^{pro} represent the power of machine i during processing and the time for machine i to process the k th operation of job j , respectively.

(3) Idle energy consumption (IEc): Refers to the energy consumption of processing equipment when it is waiting for processing, which can be expressed as:

$$IEc = \sum_{i=1}^m P_i^{idle} \times t_i^{idle} \quad (6)$$

where P_i^{idle} and t_i^{idle} represent the power and time when the machine is idle, respectively.

$$S_{jh} + t_{ijh}^{pro} + R_i \times z_i^{ojh} - L(1 - x_{ijh}) \leq C_{jh} \quad (7)$$

$$C_{jh} \leq S_{j(h+1)} \quad (8)$$

$$C_{jh_j} \leq C_{Max} \quad (9)$$

$$S_{jh} \geq r_j \quad (10)$$

$$\sum_{j=1}^n \sum_{h=1}^{h_j} x_{jh} \leq 1 \quad (11)$$

$$s_{jh} \geq 0, c_{jh} \geq 0 \quad (12)$$

$$\sum_{i=1}^m z_i^t \leq 1 \quad (13)$$

Formula (1) represents the objective function; formula (2) is the completion time; formula (3) calculates the total energy consumption of the machine; formula (4) calculates the setup energy consumption; formula (5) calculates the processing energy consumption; formula (6) calculates the idle energy consumption; formulas (7) and (8) are the operation-sequence constraints of each job, that is, the start time of an operation plus its processing time is less than the completion time of that operation, and the completion time of an operation is less than the start time of the next operation of the same job; formula (9) states that the completion time of each job cannot exceed the makespan; formula (10) states that the first operation of a job can start only after the job has arrived; formula (11) states that a machine can process only one operation at a time; formula (12) requires all decision variables to be non-negative; and formula (13) states that an operation can encounter at most one breakdown during processing.

4. Methodology

4.1. Framework

This study aims to solve the multi-objective dynamic job shop scheduling problem considering random job arrivals and machine breakdowns. The optimization objectives are to minimize the makespan and total energy consumption. Due to the high computational cost associated with evaluating candidate dispatching rules via discrete-event simulation—particularly in dynamic environments—an efficient and reliable evolutionary search framework is required. Figure 2 shows the overall framework of the algorithm. The specific process is as follows:

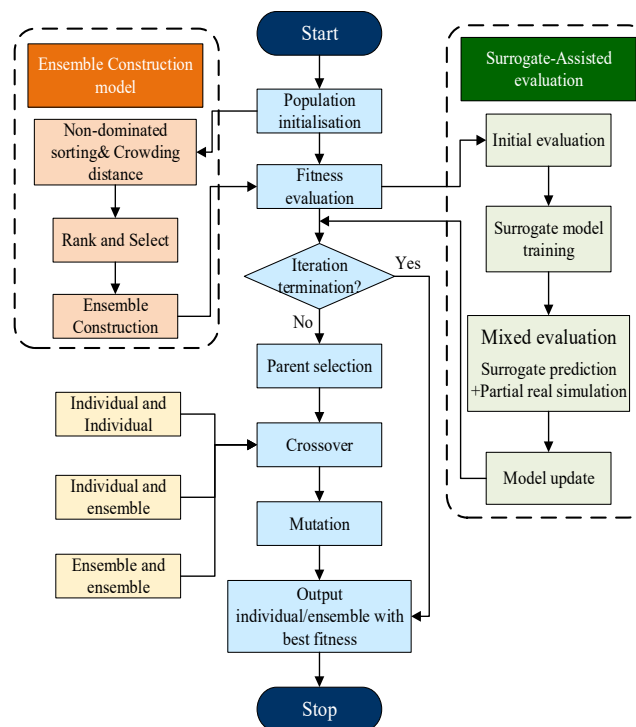


Figure 2. SA-EGP framework.

1. The initial population is generated using the ramped half-and-half method. If identical individuals (defined as two individuals with the same parse tree representation within the same population) exist, the duplicate individual is removed and a new individual is regenerated to replace it.

2. During the initialization phase, all individuals are evaluated via simulation to construct the initial training archive, which is then used for the initial training of the surrogate model.

3. All individuals are ranked based on non-dominated sorting and crowding distance.

4. Individuals with higher rankings are selected in turn to construct ensemble individuals. Each ensemble individual consists of a preset number of single individuals. After one ensemble individual is constructed, its selected members are removed from the current candidate pool, and the remaining GP individuals are used to construct the next ensemble individual, until the preset number of ensemble individuals is reached.

5. Once an ensemble individual is constructed, it no longer undergoes fitness evaluation using the traditional simulation method. Instead, it is incorporated into the evolutionary process together with single individuals and processed by the surrogate-assisted evaluation module. Specifically, for each ensemble individual, first, a structure-aware feature vector is built based on the behavioral responses of its member rules, and a complete representation is formed by combining diversity and decision conflict information; then, the trained surrogate model is used to predict its multi-objective performance. All candidate solutions are subjected to non-dominated sorting based on the predicted results. According to the sorting results, a subset of promising individuals is selected for real simulation evaluation, while the remaining individuals retain their predicted values to participate in the subsequent selection process. This approach effectively reduces computational costs while preserving evaluation accuracy.

6. Offspring for the ensemble population are generated using a new genetic operator. Unlike the traditional approach that only produces offspring through crossover and mutation between individuals, this operator supports three possible combination modes: individual–individual, individual–ensemble, and ensemble–ensemble, aiming to achieve a more diverse population.

7. The newly generated individuals and ensemble individuals are sorted and selected. Subsequently, the fitness of the new individuals is evaluated using a hybrid strategy that partially uses surrogates and partially uses real simulation. All results obtained from real simulation evaluations are used to periodically retrain the surrogate model.

8. The parent population is then updated by adding the most promising ensemble individuals and single individuals from the offspring. Meanwhile, the mechanism described in step 1 is used to remove similar individuals. If a large number of duplicates occur, the offspring batch with a lower proportion of duplicates is retained to maintain population diversity.

4.2. Individual and ensemble hybrid population

To enhance the population's performance in this dynamic multi-objective job shop context, we draw on the ensemble population method of Xu et al. [18], which has been proven to achieve stronger robustness and superior performance under dynamic and multi-scenario conditions.

The initial population is constructed using the ramped half-and-half method, yielding $individuals = \{ind_1, ind_2, \dots, ind_N\}$, where N represents the population size. Subsequently, ensemble individuals $E = \{e_{n+1}, \dots, e_N\}$ are constructed by selecting diverse and complementary individuals from the evaluated population. Specifically, the selection of ensemble members considers both Pareto dominance and differences in objective preferences. Individuals with higher non-dominated ranks are selected first to ensure good overall scheduling performance. Under the same non-dominated rank, crowding distance is used to preferentially retain individuals located in different regions of the Pareto front, thereby avoiding excessive concentration of ensemble members in similar objective-trade-off regions.

To further increase the diversity of the initial population, this paper introduces a semantic deduplication strategy. Specifically, a single scenario is configured to quickly evaluate and record the dispatch event sequence (including machine, job, operation, queue length, etc.), and the increment of the sequence is computed as the semantic fingerprint. The deduplication process consists of two levels: an expression string is used to rapidly eliminate individuals with identical structures, and semantic fingerprints are further employed to remove duplicate individuals. The eliminated individuals are

replaced by newly generated random individuals to maintain a stable population size produced by the genetic operators.

At the same time, we incorporate the new genetic operator proposed by Xu et al. [18] to enhance population diversity. The crossover and mutation operators are designed to handle both single individuals and ensemble individuals and support individual–individual, individual–ensemble, and ensemble–ensemble crossover modes, as well as two mutation modes, as illustrated in Figures 3–7. It is worth noting that the mutation operator always generates a single individual, which is then used to reform a diverse ensemble according to our construction strategy, thereby maintaining a stable proportion between single individuals and ensemble individuals.

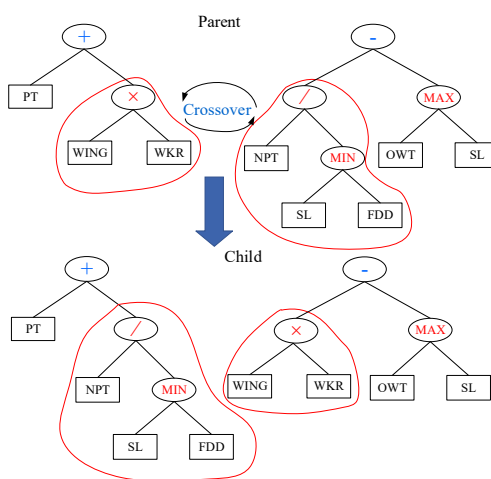


Figure 3. Individual–individual crossover.

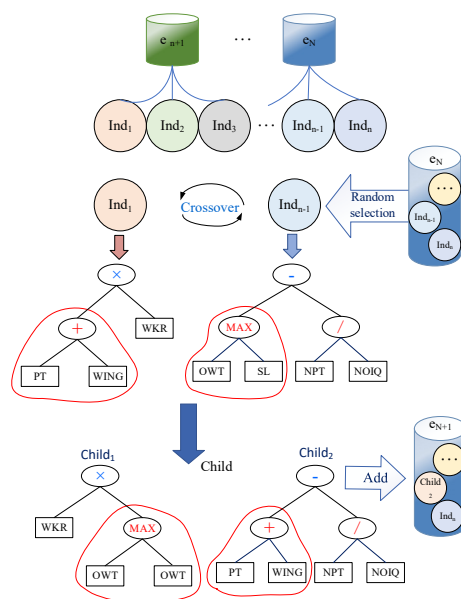


Figure 4. Individual–ensemble crossover.

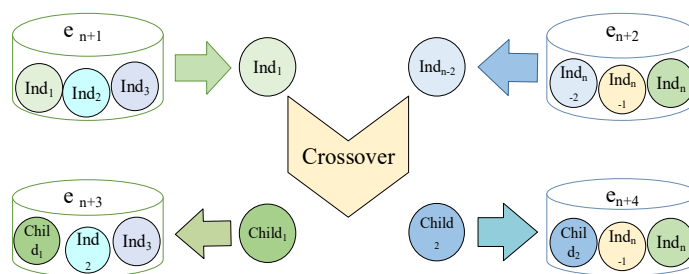


Figure 5. Ensemble-ensemble crossover.

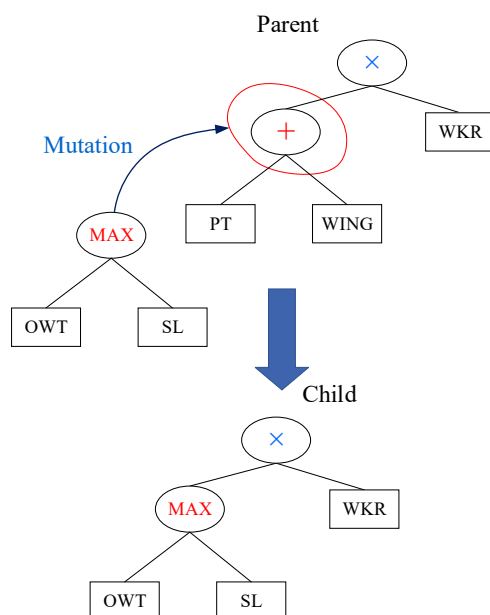


Figure 6. Individual-individual mutation.

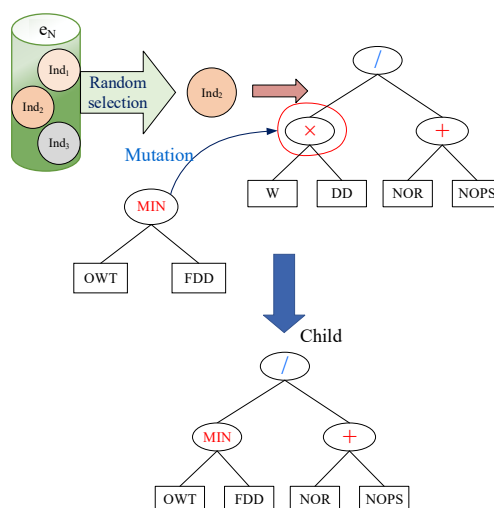


Figure 7. Ensemble-ensemble mutation.

4.3. Surrogate-assisted evaluation module

In the framework of genetic programming hyper-heuristics (GPHH), the fitness of dispatching rules is typically evaluated using discrete event simulation. This computational cost becomes particularly high when dealing with more complex ensemble individuals, because each decision point requires invoking multiple member rules.

Consequently, this cost has become the main bottleneck restricting algorithmic efficiency. Note that the surrogate is not intended to completely replace simulation-based evaluation; rather, it serves as an auxiliary estimator to accelerate evolutionary search while maintaining solution reliability.

To address this issue, this paper proposes a surrogate-assisted fitness evaluation module with uncertainty awareness, which significantly reduces computational cost while preserving evaluation accuracy. It is worth noting that the surrogate-assisted estimator is not intended to completely replace simulation-based evaluation. Instead, it serves as an auxiliary estimator to accelerate the evolutionary search while maintaining the reliability of the solutions.

4.3.1. Real fitness evaluation based on simulation

The true performance of candidate dispatching rules (whether a single individual or an ensemble) is still assessed via discrete event simulation. The simulation process begins by initializing the system clock and generating an event list, which includes job release, process arrival, and machining events. Events are then executed sequentially in time order. When a “machine selection” decision point is triggered, the current scheduling rule is invoked to prioritize the candidate operation set, thereby determining the next operation to process.

For a single individual, it directly outputs the priority value of each candidate operation. For an ensemble individual, however, the outputs from multiple dispatching rules must be combined. This paper employs a dynamic weighted voting mechanism to fuse the scores of candidate operations, as detailed in Section 4.3.2.

Although simulation-based evaluation yields accurate fitness measures for individuals, its high computational cost restricts its use to the precise assessment of only key individuals.

4.3.2. Surrogate model construction

Since dispatching rules themselves are encoded as syntax trees, the syntactic differences in tree structures cannot be directly used as numerical inputs to a surrogate model. Therefore, this paper adopts the idea of DR phenotypic representation proposed in [46], where a rule is characterized by its behavioral responses in a set of typical decision scenarios.

Specifically, the construction of the phenotypic representation takes the dispatching decision of “an idle machine selecting an operation from the queue” as the basic unit. For each decision scenario, there are several candidate operations in the queue; the rule assigns a priority value to each candidate and induces an ordering. To make the ordering results comparable across different scenarios, the phenotypic representation introduces the shortest processing time (SPT) as the baseline priority rule to generate a reference ordering, thereby producing a fixed-dimensional feature vector $\phi(x)$. Finally, the surrogate objective vector is obtained as $\hat{f}_1(x) = [C(\phi(x)), E(\phi(x))]$.

It is worth noting that the surrogate model is designed to estimate the expected performance of a rule across multiple random instances, rather than the outcome of a single simulation run. Therefore,

the surrogate model primarily supports the relative comparison of candidate solutions during the evolutionary selection process, while the final evaluation of solutions remains based on simulation.

For ensemble individuals, an ensemble individual consists of multiple dispatching rules, and its scheduling decisions depend on the synergy among its member rules. Consequently, simply aggregating the individual features of the members makes it difficult to accurately characterize the overall behavior of the ensemble. To address this issue, this paper constructs a structure-aware feature representation method for ensemble individuals. For an ensemble $E = \{x_1, x_2, \dots, x_n\}$, on the basis of extracting the individual features $\phi(x_i)$ of each member, an ensemble feature vector $\Phi_{base}(E)$ is constructed.

The phenotypic features are constructed by aggregating the decision vectors of the constituent individuals. Specifically, the decision vectors are combined using the dynamic weighted voting mechanism adopted during the simulation process [49]. During simulation scheduling, each member rule first recommends a candidate operation according to its own priority function. Its voting weight is jointly adjusted by a base weight, the current decision confidence, and a scheduling-progress factor. If the recommendation of a member rule is consistent with the final ensemble choice, its base weight is appropriately increased; otherwise, the base weight is decayed. Subsequently, the phenotypic vectors of the member rules are weighted and aggregated according to the normalized member weights, thereby obtaining a unified feature vector that represents the collective decision-making behavior of the ensemble individual.

Furthermore, in addition to the behavioral features of the ensemble, this study introduces structural supplementary features, including the diversity among individuals $D(E)$ and the degree of decision conflict $H(E)$. This representation also establishes an important distinction between surrogate modeling for single individuals and for ensembles. For a single dispatching rule, the surrogate prediction mainly depends on its behavioral responses in predefined decision scenarios. In contrast, the ensemble representation additionally incorporates member diversity and decision conflict, providing information about the interactions among the constituent rules. Therefore, the surrogate model must not only learn the mapping from behavioral features to scheduling performance but also distinguish the effects of rule interactions within an ensemble. The formulas for $D(E)$, $H(E)$, and $\Phi(E)$ are given as follows:

$$D(E) = \frac{2}{k(k-1)} \sum_{i < j} \|\phi(x_i) - \phi(x_j)\|_2 \quad (14)$$

$$H(E) = \frac{1}{T} \sum_{t=1}^T \sigma_t \quad (15)$$

$$\sigma_t = \sqrt{\frac{1}{k} \sum_{i=1}^k (r_{i,t} - \bar{r}_t)^2} \quad (16)$$

where T is the total number of decisions, k is the number of ensemble members, and σ_t denotes the standard deviation of the member performance ranks in decision scenario t . $r_{i,t}$ denotes the performance rank of members i in decision scenario t , and $\bar{r}_t$ denotes the mean performance rank of the members in decision scenario t . Finally, the feature vector of the ensemble is constructed as

$\Phi(E) = [\Phi_{base}(E), D(E), H(E)]$. The specific surrogate framework for prediction based on feature vectors is shown in Figure 8.

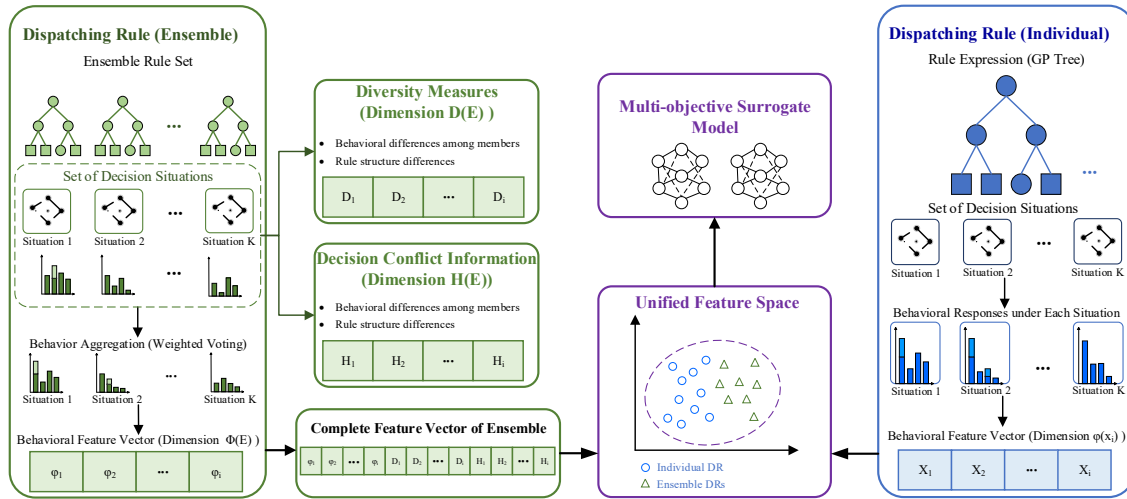


Figure 8. Unified feature representation for individual and ensemble dispatching rules.

4.3.3. Surrogate model prediction

The surrogate model is trained only on an archive consisting of solutions evaluated via real discrete-event simulation. The pseudo-code for surrogate model evaluation is provided in Algorithm 1. In the early generations, all individuals are evaluated through simulation to initialize the archive. Once the archive reaches a predefined threshold, surrogate-assisted evaluation is activated. In subsequent generations, all candidate individuals that have not yet obtained a valid fitness evaluation, including both ensemble individuals and single individuals, are predicted by the surrogate model.

In the subsequent evolutionary process, the surrogate model first predicts all individuals and forms a predicted Pareto ranking based on the predicted objective values. Then, under a limited real-evaluation budget, individuals with higher potential are selected from the prediction results for real simulation. Specifically, individuals are selected sequentially, starting from the first non-dominated front. When the number of individuals in the current non-dominated front is insufficient to meet the preset number of real evaluations, selection continues from the next non-dominated front. If the required number of real evaluations is reached within a non-dominated front, the required number of individuals is randomly selected from the remaining individuals in that front, until the number of individuals requiring real evaluation, N_{real} is reached.

In ensemble surrogate-assisted evaluation, surrogate models may not always provide reliable fitness predictions, especially in early generations or under highly dynamic scheduling conditions. Applying a fixed surrogate evaluation directly to all individuals not only slows down the training of surrogate models but may also introduce misleading data, potentially deteriorating the evolution of the population. Therefore, a dynamic evaluation ratio is needed to adaptively balance surrogate-based evaluation and actual simulation.

Specifically, according to the non-dominated sorting results, the top $p\%$ of individuals are selected for real evaluation, while the remaining individuals use the surrogate-predicted values in the

subsequent evolutionary process. The evaluation ratio p is dynamically adjusted with the generation number: it remains high in the early stages of evolution to enhance search capability and gradually decreases in later stages to improve computational efficiency. The specific formulas for calculating the evaluation ratio are as follows:

Algorithm 1: Surrogate-assisted Fitness Evaluation

Input: Population U ; Archive $H = \{\Phi(x), f(x)\}$; Surrogate models M ;
Evaluation ratio $p(t)$.
Output: Estimated fitness F_{est} .

```

1  if  $|H| < H_{min}$  then
2      for each  $x \in U$  do
3          Compute true fitness:
4           $f(x) = (C_{max}(x), E(x))$ 
5          Extract feature vector  $\Phi(x)$ :
6           $H \leftarrow H \cup \{(\Phi(x), f(x))\}$ 
7      end
8      Train  $M_1$ , and  $M_2$  on  $H$ 
9      Return  $F_{est} = f(x)$ 
10 end
11 //Surrogate prediction
12 for each  $x \in U$  do
13     Extract feature vector  $\Phi(x)$ 
14     Predict objectives:
15      $\hat{C}_{max}(x) = M_1(\Phi(x))$ 
16      $\hat{E}(x) = M_2(\Phi(x))$ 
17     Set predicted fitness:
18      $\hat{f}(x) = (\hat{C}_{max}(x), \hat{E}(x))$ 
19 end
20 //Ranking based on predicted fitness
21 Compute non-dominated sorting:
22      $Rank(x) = R(\hat{f}(x))$ 
23 Compute crowding distance:
24      $CD(x) = C(\hat{f}(x))$ 
25 Determine evaluation ratio  $p = p(t)$ 
26  $N \leftarrow |U|$ 
27 Select top  $p \times N$  solutions based on  $(Rank, CD)$ , denoted as  $S_{eval}$ 
28 for each  $x \in S_{eval}$  do
29     Compute true fitness  $f(x)$ 
30     Replace  $\hat{f}(x) \leftarrow f(x)$ 
31      $H \leftarrow H \cup \{(\Phi(x), f(x))\}$ 
32 end
33 Retrain  $M_1, M_2$  using  $H$  periodically
34 Return  $F_{est} = f(x) \cup \hat{f}(x)$ 

```

$$t = \frac{g}{G} \quad (17)$$

$$p_{progress} = p_{min} + (p_{max} - p_{min})t^2 \quad (18)$$

$$p_{accuracy} = p_{min} + (p_{max} - p_{min})\max(0, \bar{\rho}_g) \quad (19)$$

$$\bar{\rho}_g = \beta \rho_g + (1 - \beta)\bar{\rho}_{g-1} \quad (20)$$

$$p_g = \alpha p_{progress} + (1 - \alpha)p_{accuracy} \quad (21)$$

Here, g denotes the current generation, G denotes the total number of iterations, $p_{progress}$ represents the surrogate evaluation ratio determined by evolutionary progress, $p_{accuracy}$ represents the surrogate evaluation ratio determined by the ranking prediction ability of the surrogate model, α denotes the trade-off coefficient between the evolutionary-progress factor and the surrogate-ranking-consistency factor, β denotes the exponential moving average (EMA) smoothing coefficient, and ρ_g denotes the Spearman rank correlation coefficient between the surrogate model's predicted values and the true evaluation values in the g -th generation. $\bar{\rho}_g$ denotes the smoothed value of the surrogate ranking in the g -th generation. p_{min} and p_{max} denote the lower and upper bounds of the surrogate evaluation ratio, respectively. p_g denotes the surrogate evaluation ratio actually adopted in the g -th generation. The preset parameters are given in Table 2.

Table 2. Dynamic evaluation parameters.

Symbol	Parameter setting
p_{min}	0.2
p_{max}	0.85
α	0.5
β	0.3
$\dot{g}$	5
H_{min}	50

Table 3. Sensitivity of SA-EGP to the fusion weight α and the surrogate retrain interval $\dot{g}$.

Parameter	HV	IGD	Real evaluations
$\alpha = 0.25$	0.972 ± 0.013	0.017 ± 0.010	8355
$\alpha = 0.5$	0.973 ± 0.011	0.013 ± 0.004	8278
$\alpha = 0.75$	0.967 ± 0.023	0.016 ± 0.014	7978
$\dot{g} = 5$	0.972 ± 0.011	0.012 ± 0.003	8278
$\dot{g} = 10$	0.968 ± 0.012	0.014 ± 0.005	8475
$\dot{g} = 20$	0.969 ± 0.014	0.022 ± 0.023	8554

We also tested two key parameters, α and $\dot{g}$. The results show that when α is set to 0.25, 0.5, and 0.75, no significant difference is observed. Therefore, we select 0.5, which performs relatively better in the average results of 10 repeated experiments, as the default value; similarly, the default value of g is set to 5. Parameter sensitivity data are given in Table 3.

5. Experimental design

In this chapter, we provide a detailed description of the settings for subsequent experiments. First, we introduce the parameter configuration of DJSS, followed by the terminal set and function set of the GP designed in this paper. Second, we introduce the types of comparative models. Finally, we present the evaluation metrics for the bi-level surrogate model algorithm used in this paper, namely the hypervolume metric and the inverted generational distance.

(1) For the workshop parameters, we adopted the commonly used workshop parameter configuration of the surrogate model [14]. A total of 2500 jobs are processed on 10 machines, with the first 500 jobs serving as warm-up jobs and not included in the calculation. The number of operations for each job is uniformly sampled within a given interval (such as 2–10 processes), and the standard processing time for each operation follows a uniform distribution [1,49].

The machine breakdown rate follows an exponential distribution, while the machine maintenance time follows a uniform distribution. These machines are located at fixed positions in the workshop, and transportation time is ignored.

The arrival process of jobs follows a Poisson process, with its arrival rate jointly determined by the expected utilization rate of the system, average processing time, average number of processes, and number of machines [50]. It satisfies:

$$b = \frac{1}{\lambda} = \frac{\mu_p \mu_g}{Um} \quad (22)$$

In the formula, λ represents the average arrival rate of jobs (i.e., the number of new jobs arriving at the workshop per unit time), which depends on the average processing time of jobs μ_p , the average number of operations for jobs μ_g , the system utilization rate U , and the number of machines m in the workshop.

In the simulation, utilization rate is an important factor reflecting the busyness of the job shop. A higher utilization rate indicates that the machines are busier. In this work, we consider two different levels of utilization rates (0.80 and 0.90). The parameters of the job shop model are shown in Table 4.

Table 4. Model parameters.

Parameter	Value
Machine	10
Job	2500
Operation	[2, 10] uniform distribution
Processing time	[1, 49] uniform distribution
Utilization rate	(0.80, 0.90)

(2) GP parameters

In this study, the commonly used terminal and function sets in GP algorithms are adopted, and terminal sets related to machine breakdowns and energy consumption are designed according to the DJSS, as shown in Table 5.

Table 5. Terminal set and function set.

Symbol	Description
PT	The processing time of the current operation for the job
RPT	The total processing time for all remaining processes of the job
RO	Number of remaining processes for the job
AT	Job arrival time
WT	The waiting time for the current process of this job
IDLE	Current machine idle time
MTTR	Average repair time of all completed breakdowns of the current machine up to the current moment
MQL	Current machine queue length
MWL	Current machine workload = remaining setup time + remaining processing time
PE	Cumulative processing energy consumption of the current machine
IE	Cumulative idle energy consumption of the current machine
CMF	Cumulative number of breakdowns of the current machine
ADD	Addition
SUB	Subtraction
MUL	Multiplication
DIV	Division
MAX	Maximum
MIN	Minimum
IFTE	Logical operators

In the experiments, the ramped half-and-half method is used to generate 200 individuals, as the initial population. The maximum number of generations is set to 50, and the crossover and mutation rates are set to 90% and 10%, respectively. Meanwhile, the size of the offspring population generated is set to 200. The remaining parameters can be found in Table 6.

Table 6. Population parameter.

Parameter	Value
Iteration count	50
Population size	200
Initial population	Ramped half-and-half (depth 2–6)
Selection	Non-dominated sorting
Crossover	0.9
Maximum depth of crossover	17
Mutation rate	0.1
Maximum mutation depth	7

(3) Surrogate-assisted model

To verify the superiority of the surrogate model adopted in this paper, seven common machine learning models are selected for comparison: linear regression (LR), decision tree (DT), random forest (RF), gradient boosting (GB), support vector machine (SVM), K-nearest neighbors (KNN), and multilayer perceptron (MLP).

The above machine learning models are implemented using the scikit-learn library in Python [51]. It is worth noting that the purpose of this paper is to find a surrogate model suitable for EGP, rather than to perform additional parameter tuning for each model. The MLP is implemented in PyTorch, and its specific parameters are listed in Table 7.

Table 7. MLP parameters.

Parameter	Setting
Training epochs	150 (stop if no improvement for 15 consecutive epochs)
Optimizer	Adam
Learning rate	0.001
Weight decay	0.01
Dropout rate	0.4
Batch size	64
Input layer size	32
Output layer size	1
Hidden layer sizes	64 and 32 (two layers)

(4) Algorithm evaluation metrics

To evaluate the performance of the algorithm, this study adopts the most widely used metric in multi-objective optimization, namely the hypervolume (HV) indicator, which measures algorithm performance by quantifying the objective space size occupied by the non-dominated solutions. A larger HV indicates better algorithm performance (i.e., higher overall quality of the found solutions). The specific calculation formulas are given as follows:

$$HV(P, U) = Volume \left(\bigcup_{p \in P} \prod_{i=1}^m [f_i(p), U_i] \right) \quad (23)$$

where P is the set of non-dominated solutions to be evaluated, and U is the reference point, set as $U = (1.1 \max C_{max}, 1.1 \max E_{total})$.

Of course, HV alone does not fully reflect the performance of the algorithm. Therefore, to mitigate the possibility that some algorithms may have an undue advantage in terms of HV, the inverted generational distance (IGD) is also introduced as an evaluation metric.

$$IGD(PF, PF^*) = \frac{1}{|PF^*|} \sum_{y \in PF^*} \min_{x \in PF} d(x, y) \quad (24)$$

where PF is the set of non-dominated solutions to be evaluated, PF^* is the reference Pareto front formed by the best non-dominated solutions obtained by all algorithms, $|PF^*|$ is the number of points in the reference front, and $d(x, y)$ denotes the Euclidean distance between solution x and reference point y :

$$d(x, y) = \sqrt{\sum_{i=1}^m (f_i(x) - f_i(y))^2} \quad (25)$$

6. Experimental results

(1) Ensemble size test analysis

The impact of different ensemble sizes on the overall quality and convergence speed of the population is investigated by evaluating the algorithm performance under four different scenarios (ensemble sizes of 5, 10, 20, and 30) over 10 repeated independent runs. Of course, the total population size remains unchanged; when the ensemble size is increased from 5 to 10, the corresponding number of individuals in the population decreases by 5.

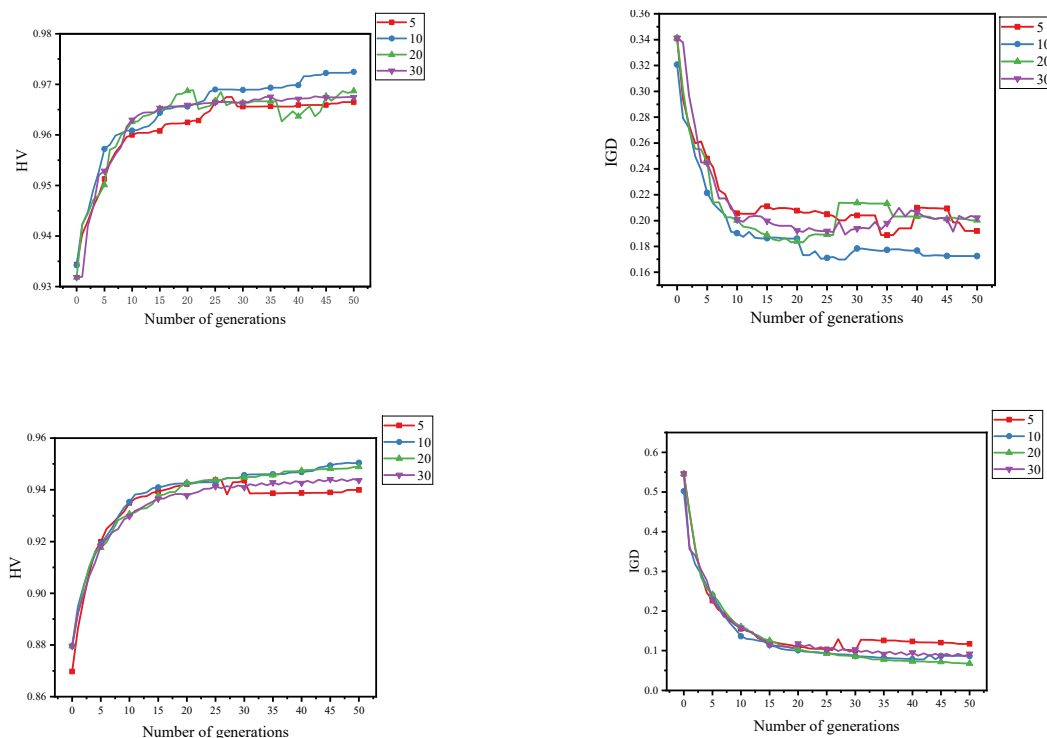


Figure 9. Comparison of convergence performance for different ensemble numbers of HV and IGD under utilization rates of 0.8 and 0.9.

As shown in Figure 9, for the two different utilization rates, an ensemble size of 10 achieves better performance under both utilization rates, followed by ensemble sizes of 20, 30, and 5. It is worth noting that when the ensemble size is 20, the IGD value under a utilization rate of 0.9 gradually surpasses that of the ensemble size of 10 after the 30th generation. However, compared with an ensemble size of 20, an ensemble size of 10 exhibits better convergence speed and performance in all other scenarios. Therefore, we set the ensemble size to 10 for further analysis in subsequent experiments.

(2) Surrogate model evaluation

The accuracy of fitness prediction by the surrogate model directly determines the quality of the surrogate model. Different surrogate models, due to their distinct algorithmic mechanisms, may yield varying prediction accuracies. Therefore, to identify the most suitable surrogate model for fitness prediction, multiple machine learning models are compared, including LR, DT, RF, GB, SVM, KNN, and MLP. These models are trained using phenotypic feature vectors extracted from evaluated individuals.

A total of 10 independent experiments were conducted in this study. A population consisting of 200 individuals, including 10 ensemble individuals, was evolved for 50 generations, after which the 100 best-performing individuals were selected as the basis for evaluating model accuracy. Meanwhile, we observed that the change in fitness approaches a nearly linear trend in the later stages of evolution; therefore, simply using the fitness value as the sole criterion for model accuracy does not reflect true predictive performance. Consequently, the accuracy assessment in this experiment considers two aspects: the accuracy of fitness prediction and the difference between the ranking of individuals based on predicted fitness and that based on actual evaluated fitness. This dual-aspect evaluation aims to reveal a more realistic picture of model prediction performance. The ranking-consistency indicator can further reflect the potential impact of surrogate prediction error on the evolutionary selection process, so that the model prediction can be evaluated in a more realistic manner. The specific calculation formula is as follows:

$$NRD_i = 1 - \frac{2D}{N(N-1)} \quad (26)$$

Here, D represents the logarithm of individuals with inconsistent rankings, that is, the comprehensive ranking scores S_i , $\hat{S}_i$ are calculated based on non-dominated solutions for the real target and the predicted target, respectively. The number of individuals with different scores for S_i and $\hat{S}_i$ is counted as D . N is the number of individuals in the test set. The closer NRD is to 1, the more accurate the predicted ranking is.

$$ACC_i = 1 - \sum_{i=1}^N \left| \frac{|\hat{C}_i - C_i|}{R_C} \right| + \left| \frac{|\hat{E}_i - E_i|}{R_E} \right| \quad (27)$$

Here, R_C and R_E predict the range of two individual targets, $\hat{C}_i$ and C_i represent the predicted completion target value and the actual completion target value, respectively, and $\hat{E}_i$ and E_i represent the predicted energy consumption target value and the actual energy consumption target value, respectively. The closer ACC is to 1, the more accurate the predicted fitness is.

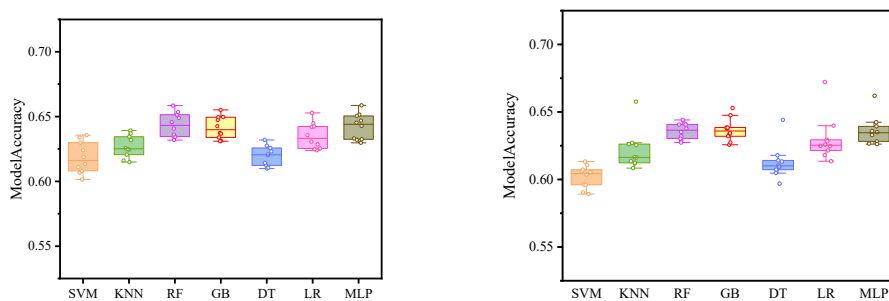


Figure 10. Selection accuracy from 10 independent runs under two utilization rates (0.8 and 0.9).

Table 8. Prediction accuracy of different machine learning models at two different utilization rates.

Utilization	Model	Mean	Std	Min	Max
0.8	SVM	0.618276558	0.012060572	0.601407049	0.635661656
0.8	KNN	0.626923709	0.008474991	0.614876091	0.639224404
0.8	RF	0.643562903	0.009352391	0.631821385	0.65849522
0.8	GB	0.64142218	0.008625017	0.630990851	0.655184027
0.8	DT	0.619647742	0.007638752	0.609898408	0.631948456
0.8	LR	0.63509843	0.009986527	0.623906815	0.652827445
0.8	MLP	0.642296257	0.009969911	0.629701182	0.65868541
0.9	SVM	0.601787041	0.008348418	0.589221145	0.613254803
0.9	KNN	0.621533434	0.014280317	0.608437122	0.657591397
0.9	RF	0.635864312	0.005917602	0.627309334	0.644119203
0.9	GB	0.636804888	0.008376176	0.625740744	0.65294436
0.9	DT	0.612786614	0.012400694	0.596887476	0.644136222
0.9	LR	0.629802787	0.01645718	0.613431197	0.672153877
0.9	MLP	0.636266353	0.010474222	0.626178419	0.662024453

As shown in Table 8 and Figure 10, the overall distribution indicates that the prediction accuracy of all models in both scenarios falls within the range of 0.5–0.7, suggesting that the constructed surrogate models can, to a certain extent, reflect the actual simulation evaluation results.

Specifically, in terms of accuracy, both RF and GB demonstrate high performance. The inputs used in this study are not simple low-dimensional statistical features, but rather high-dimensional phenotypic features composed of behavioral responses to rules across 100 typical decision scenarios, where priority is given to scenarios such as machine breakdowns. Furthermore, the ensemble models incorporate structural information such as diversity and decision conflicts; consequently, the relationship between the objective function and the input features exhibits distinct nonlinearity, interactivity, and noise characteristics. Against this backdrop, tree-based ensemble models such as random forest and gradient boosting can effectively capture complex piecewise nonlinear relationships through the combination of multiple decision trees. They maintain strong adaptability to local patterns and variable interactions within high-dimensional features, while also exhibiting good robustness against simulation noise and sample perturbations; MLPs, leveraging their nonlinear mapping capabilities, can further uncover the complex functional relationships between phenotypic traits and scheduling performance, thereby maintaining strong competitiveness in terms of accuracy metrics.

In contrast, models such as linear regression are constrained by the linearity assumption and struggle to capture the complex mapping between behavioral traits and true fitness; KNN is prone to distance degradation issues in high-dimensional spaces; SVM possesses nonlinear modeling capabilities, being more sensitive to parameter settings and changes in sample distribution; and DT, on the other hand, can learn nonlinearities but is susceptible to perturbations in the current training samples due to excessive variance.

One of the objectives of this study is to identify a suitable surrogate model for this approach, which combines ensemble learning with genetic programming, so as to accelerate the fitness evaluation simulation process within an acceptable error margin. This implies that the surrogate model should not only predict simulation results but also minimize computational costs as a key consideration.

The total simulation evaluation time includes not only the actual time required to predict the fitness of the offspring but also the training time of the surrogate model. If a surrogate model has high prediction accuracy but incurs excessive training costs, its overall acceleration effect may be offset; therefore, a balance must be struck between prediction accuracy and computational cost when selecting a surrogate model.

To verify this advantage, this paper uses seven different machine learning models—LR, DT, RF, GB, SVM, KNN, and MLP—with their real evaluated runtime as the baseline. We compare the total runtime ratios during the surrogate-assisted evaluation process under two different utilization rates, conducting 10 rounds of simulation experiments.

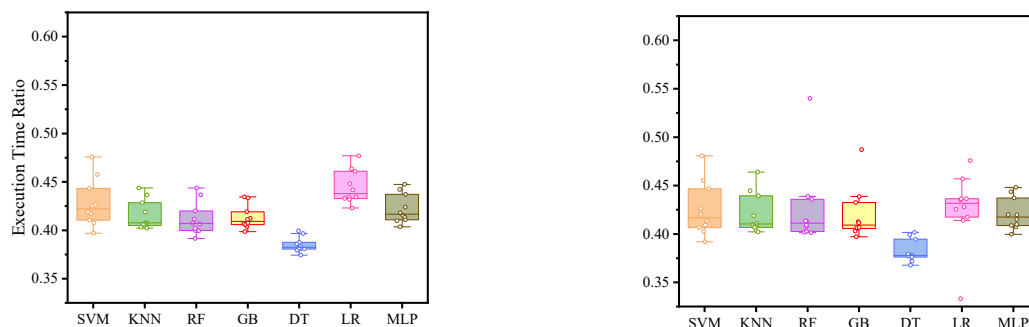


Figure 11. Ratio of model execution times based on full evaluation at utilization rates of 0.8 and 0.9.

Table 9. Ratio of complete evaluation time based on different models at different utilization rates.

Utilization	Model	Mean	Std	Min	Max
0.8	SVM	0.427950772	0.02422	0.396883	0.475695062
0.8	KNN	0.415891565	0.015169	0.402194	0.443712122
0.8	RF	0.411796581	0.016861	0.39148	0.443673931
0.8	GB	0.413285842	0.012171	0.398631	0.434519448
0.8	DT	0.38490698	0.007803	0.374243	0.399389909
0.8	LR	0.444722684	0.017242	0.422943	0.476881089
0.8	MLP	0.422114123	0.014988	0.403509	0.44730236
0.9	SVM	0.425259542	0.027586	0.391919	0.480655844
0.9	KNN	0.420796343	0.02103	0.402132	0.463843855
0.9	RF	0.426354936	0.042059	0.40131	0.539936438
0.9	GB	0.420118449	0.026927	0.397065	0.487138578
0.9	DT	0.382250087	0.01182	0.367675	0.401595266
0.9	LR	0.425848421	0.03742	0.333069	0.475860809
0.9	MLP	0.420862085	0.016546	0.399656	0.448045804

As shown in Figure 11 and Table 9, from the overall results, there is a significant difference in computational efficiency between different surrogate models in the two scenarios; especially, as the simulation scale increases, the different surrogate models can save computational costs and significantly increase them.

The DT model showed the lowest running time, indicating that, due to its relatively simple structure and low computational complexity in training and prediction processes, it has high computational efficiency in the overall evaluation process. However, due to the limited fitting ability of linear models and single decision trees to complex nonlinear relationships, their prediction accuracy is relatively low; thus, certain limitations remain in proxy modeling tasks.

In contrast, the running time of learning models such as KNN, RF, and GB is at a moderate level, indicating that these four models have achieved a certain balance between computational complexity and predictive ability, but their overall efficiency is still inferior to simpler structured models. However, LR, SVM, and MLP exhibit poor computational efficiency.

From the comprehensive prediction accuracy experiment and running time experiment, it can be seen that there is a clear trade-off between the prediction ability and computational cost of different proxy models. In practical applications, a comprehensive balance needs to be struck between prediction accuracy and computational efficiency. Therefore, based on the experimental results of this article, we selected two performance models, GB and RF, which have good predictive performance and computational efficiency, for further analysis in subsequent experiments. They are referred to as EGPHH_GB and EGPHH_RF, respectively.

(3) Ablation experiment

To examine whether the proposed dynamic evaluation ratio allocates real simulations more effectively than a fixed ratio, we compare the dynamic evaluation mechanism with a fixed evaluation mechanism and observe the prediction accuracy of population individuals and the achieved real evaluation ratio in each generation.

As shown in Figure 12 and Table 10, both fixed-ratio evaluation and dynamic evaluation can maintain usable surrogate prediction accuracy during evolution, indicating that, under both mechanisms, the GB surrogate can utilize the real evaluation samples continuously written into the training archive to learn the mapping between individual phenotypes and scheduling performance.

Specifically, in the early stage of evolution, training samples are limited. With a fixed $p = 0.5$, only about half of the offspring undergo real evaluation in each generation, so the surrogate model sees relatively few new real labels, and its accuracy recovers slowly after a decline. In contrast, dynamic evaluation lowers the surrogate ratio in the early stage of population evolution, allowing more individuals to enter real simulation and the archive, thereby correcting the early mapping more quickly. In the middle and late stages, as the population structure becomes more similar, the surrogate ratio is gradually increased, so that the limited simulation budget is reserved for more informative individuals. Compared with a fixed evaluation ratio, the dynamic evaluation ratio can arrange the proportion of real simulation and surrogate evaluation more reasonably according to the evolutionary stage: a higher real-evaluation ratio is used in the early stage to accelerate archive updating, and a higher surrogate ratio is used in the later stage to save simulation. This dynamic evaluation strategy effectively changes the allocation of real evaluations and brings higher short-term accuracy in the early stage when the budget is tight.

Furthermore, to demonstrate that our proposed multi-dimensional feature vector $\Phi(E) = [\phi(E), D(E), H(E)]$ facilitates better training of the surrogate model compared with using $\phi(E)$ alone, we conducted comparative experiments by feeding these two types of feature representations separately into the surrogate model and observing the prediction accuracy for ensemble individuals in each generation of population evolution.

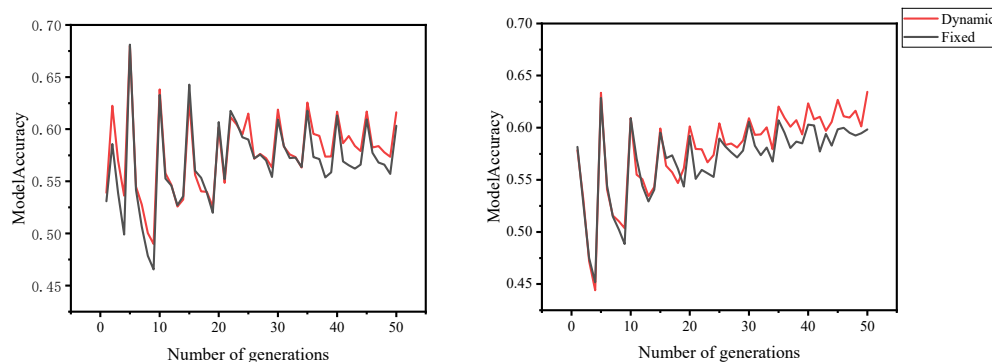


Figure 12. Variation of surrogate model accuracy with the number of generations under dynamic and fixed evaluation ratios (GB and RF, from left to right).

Table 10. Final accuracy of the surrogate model under fixed and dynamic evaluation ratios.

run	GB_fixed	GB_Dynamic	RF_fixed	RF_Dynamic
1	0.600846395	0.622196719	0.63772543	0.668024
2	0.610682642	0.631828329	0.663763427	0.641524
3	0.623116182	0.619159483	0.691725711	0.521559
4	0.600989974	0.608841282	0.526282211	0.694549
5	0.59529766	0.606091889	0.580241075	0.537067
6	0.6024232	0.599620924	0.570304186	0.69288
7	0.598318023	0.62771435	0.550432483	0.725444
8	0.590327025	0.611335368	0.565773672	0.610157
9	0.605780214	0.61628098	0.604475205	0.611894
10	0.605889339	0.610760195	0.593174343	0.640256
mean	0.603367066	0.615382952	0.598389774	0.634336

From Figure 13 and Table 11, it can be observed that both feature representations can gradually improve the prediction accuracy of the surrogate model, with the continuous accumulation of real evaluation samples. The overall prediction error shows a decreasing trend as evolution progresses, indicating that the surrogate model can progressively learn the mapping relationship between ensemble individuals and scheduling performance by continuously updating the training archive. However, compared with the phenotypic behavioral feature representation using only $\phi(E)$, the multi-dimensional feature representation $\Phi(E)$ proposed in this paper exhibits faster convergence speed and higher prediction stability throughout the entire evolutionary process.

From the perspective of error variation trends, the surrogate model employing $\Phi(E)$ can reach a lower prediction error more quickly in the early stage of evolution and maintains a relatively stable trend in subsequent iterations. In contrast, when only $\phi(E)$ is used, the model often requires more training samples to achieve a comparable accuracy and still exhibits noticeable fluctuations in the mid-to-late stages. This indicates that the structure-aware representation proposed in this paper not only improves the prediction accuracy of the surrogate model for ensemble individuals but also enhances the stability and generalization ability of the model training process.

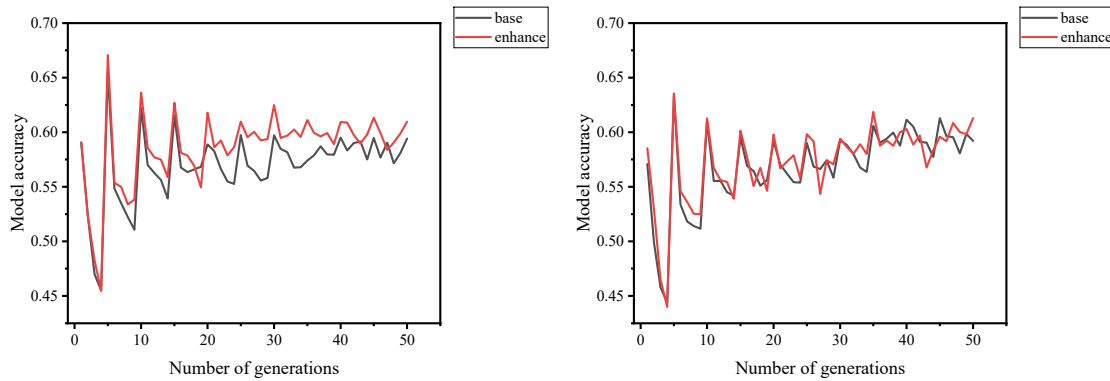


Figure 13. Accuracy variation curves of ensemble individuals under two feature representations (GB and RF, from left to right).

Table 11. Prediction accuracy after 50 iterations between GB and RF models.

run	GB_base	GB_enhance	RF_base	RF_enhance
1	0.577918721	0.598497525	0.604089777	0.692966802
2	0.613959525	0.708527157	0.605769007	0.608404346
3	0.639453183	0.647489604	0.519806027	0.556345258
4	0.612263861	0.583153758	0.561882211	0.661549407
5	0.613154844	0.584628962	0.614408323	0.553731892
6	0.623884648	0.531561466	0.557978124	0.56549269
7	0.549189567	0.555779418	0.599272444	0.681998809
8	0.499168269	0.59652007	0.619087016	0.578861034
9	0.577167197	0.591772445	0.578550838	0.580154533
10	0.635328102	0.609147821	0.659682224	0.569025987
mean	0.594148792	0.600707823	0.5920526	0.60485308

Specifically, in the early stage of evolution, due to the limited number of surrogate model training samples, the model using only $\phi(E)$ relies primarily on the average behavioral response of ensemble individuals for prediction. Although this can capture the decision-making tendencies of the ensemble in typical scheduling scenarios to some extent, it fails to reflect the structural differences and collaborative relationships among the ensemble members. Consequently, the prediction error decreases at a relatively slow rate. After incorporating the diversity feature $D(E)$ and the decision conflict feature $H(E)$, the surrogate model obtains richer structural information, enabling it not only to perceive the behavioral output of the ensemble as a whole but also to further identify the degree of difference among member rules and the level of decision consistency, thereby improving the expressive power of ensemble scheduling behavior.

In summary, compared with the feature representation based solely on behavioral responses, $\Phi(E)$, which integrates behavioral information, diversity information, and decision conflict information, can more comprehensively describe the structural and decision-making characteristics of ensemble individuals. It provides a more discriminative input representation for the surrogate model, thus effectively enhancing its ability to predict the fitness of ensemble individuals.

(4) Algorithm performance testing

To demonstrate the effectiveness of the proposed SA-EGP, we compared its performance with several representative scheduling methods through 10 repeated experiments, including standard GP, SA-GP_GB, SA-GP_RF[12], and EGP, as well as SA-EGP_GB and SA-EGP_RF. To ensure fairness among different algorithms, we adopted a different metric from the iteration count, namely the evaluation frequency metric. Since surrogate-assisted algorithms incur lower simulation computational costs for the same number of iterations, the number of evaluations per algorithm under the same iteration may vary. Therefore, it is meaningful to use the evaluation frequency indicator to ensure fairness.

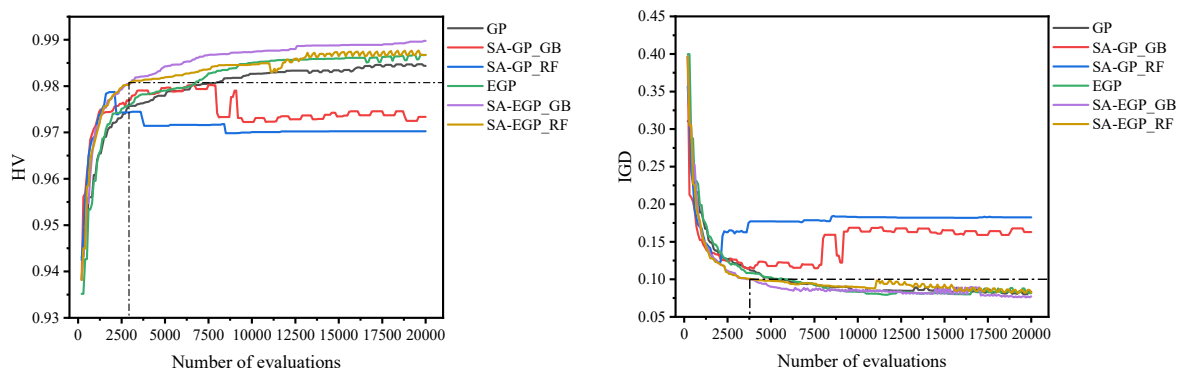


Figure 14. Convergence curves of HV and IGD with respect to the number of evaluations under different algorithms.

In Figure 14, the convergence of the six different algorithms over 20,000 evaluations is presented. First, it is clear that SA-EGP_GB and SA-EGP_RF, which incorporate surrogate models, exhibit faster convergence than EGP. At around 2700 evaluations, both algorithms achieve the convergence performance that EGP attains at 6250 evaluations, demonstrating excellent convergence speed. However, as the number of evaluations increases, the convergence of all algorithms gradually stagnates. Moreover, compared with EGP, both SA-EGP_GB and SA-EGP_RF show better overall performance, indicating that the description of individual phenotypic features indeed contributes to algorithm convergence, a finding that has been confirmed by Zeiträg et al. [12].

To further verify the stability of the algorithms over multiple independent runs and whether the performance differences are statistically significant, the final HV and IGD of each algorithm over 10 independent runs were recorded (as shown in Tables 12 and 13), and the Wilcoxon signed-rank test was conducted at a significance level of 0.05. The test results in Tables 14 and 15 show that, in terms of HV, SA-EGP_GB is significantly better than GP, SA-GP_GB, and SA-GP_RF ($P < 0.05$), while the differences from EGP and SA-EGP_RF are not significant; in terms of IGD, SA-EGP_GB is significantly better than SA-GP_GB and SA-GP_RF ($P < 0.05$), while the differences from GP, EGP, and SA-EGP_RF are not significant. This indicates that the introduction of the surrogate model mainly improves the early search efficiency and overall convergence quality, while the original EGP remains strongly competitive in final population diversity.

Table 12. Comparison of HV between 10 independent runs of different algorithms.

Run	GP	SA-GP_GB	SA-GP_RF	EGP	SA-EGP_GB	SA-EGP_RF
1	0.982898816	0.962909436	0.951122575	0.989979445	0.990182756	0.98529277
2	0.98359478	0.980761262	0.983980333	0.986242894	0.989076324	0.986446436
3	0.982367762	0.913967427	0.972640738	0.984839796	0.992291857	0.993228141
4	0.989319588	0.97677828	0.97574643	0.991729312	0.991247634	0.986827585
5	0.981873	0.985692394	0.967729639	0.983969254	0.995786548	0.987057149
6	0.981000319	0.989828123	0.967397146	0.983653593	0.986548995	0.992338448
7	0.988163456	0.976444426	0.990763844	0.993185149	0.987988454	0.972558856
8	0.988819159	0.976258182	0.926167588	0.990586796	0.990638911	0.991482003
9	0.982742399	0.983602781	0.988074412	0.982251201	0.988793072	0.985837914
10	0.986565298	0.978779327	0.978786931	0.983368432	0.984965534	0.98699431
mean	0.984734458	0.972502164	0.970240964	0.986980587	0.989752009	0.986806361

Table 13. Comparison of IGD with 10 independent runs between different algorithms.

Run	GP	SA-GP_GB	SA-GP_RF	EGP	SA-EGP_GB	SA-EGP_RF
1	0.080066897	0.185637175	0.274780033	0.069178164	0.071351573	0.100090103
2	0.095366624	0.143833174	0.10050793	0.068182606	0.073059179	0.06631413
3	0.075259706	0.187487326	0.1768273	0.093541671	0.054790487	0.064392325
4	0.065078296	0.118674518	0.124280296	0.103787217	0.066310586	0.079570709
5	0.083119821	0.089294593	0.223899409	0.110615474	0.052612922	0.097067133
6	0.086833357	0.075951886	0.172814405	0.084724213	0.119575969	0.06192735
7	0.088375528	0.118489867	0.058986075	0.044909885	0.10240337	0.122189007
8	0.083910249	0.126785783	0.454094466	0.071622098	0.064286904	0.062452418
9	0.084187259	0.076648232	0.104505658	0.081799736	0.08538869	0.078580154
10	0.062658734	0.140728157	0.134874303	0.076974993	0.070151052	0.085994453
mean	0.080485647	0.126353071	0.182556988	0.080533606	0.075993073	0.081857778

Table 14. Wilcoxon signed-rank test results based on HV and IGD (baseline algorithm: SA-EGP_GB).

Comparison Algorithm	Metric	P-value	Significant ($\alpha = 0.05$)
GP	HV	0.0195	Yes
	IGD	0.556	No
SA-GP_GB	HV	0.0039	Yes
	IGD	0.002	Yes
SA-GP_RF	HV	0.0098	Yes
	IGD	0.002	Yes
EGP	HV	0.0645	No
	IGD	0.492	No

Table 15. Wilcoxon signed-rank test results based on HV and IGD (baseline algorithm: SA-EGP_RF).

Comparison Algorithm	Metric	P-value	Significant ($\alpha = 0.05$)
GP	HV	0.105	No
	IGD	0.556	No
SA-GP_GB	HV	0.0039	Yes
	IGD	0.002	Yes
SA-GP_RF	HV	0.0039	Yes
	IGD	0.002	Yes
EGP	HV	0.845	No
	IGD	0.556	No

Meanwhile, we notice that SA-EGP_RF is overtaken by EGP at around 11,000 evaluations. We suspect this may still be due to the selection accuracy of the surrogate model. It is worth noting that the performance of SA-GP_GB and SA-GP_RF is lower than that of GP and EGP, while the prediction accuracy of GB and RF themselves is not poor. Therefore, this performance degradation cannot be simply attributed to the regression ability of the surrogate model. A more likely reason lies in the coupling between surrogate-assisted selection and the search structure of single-rule GP. In single-rule GP, the surrogate prediction results directly affect individual ranking and the selection of real evaluation samples. Prediction errors may cause high-quality rules to be underestimated or low-quality rules to be retained and further affect subsequent evolutionary search, gradually biasing the search toward a limited region of the solution space. The sparse and weakly continuous Pareto fronts of SA-GP_GB and SA-GP_RF in Figure 14 are also consistent with this phenomenon.

Overall, the surrogate model mainly serves to improve early-stage search efficiency, especially when combined with EGP, enabling a rapid attainment of a high-quality solution set with fewer evaluations. However, as the number of evaluations increases, the approximation error introduced by the surrogate model may gradually affect the accuracy of individual selection. Nevertheless, with more simulation evaluations, both SA-EGP_RF and SA-EGP_GB, which employ surrogate models, exhibit performance

no worse than the original algorithm. Therefore, using a surrogate model that aligns with the corresponding algorithm not only accelerates the iterative evolution of the population but also helps the algorithm explore the Pareto front more comprehensively under scenarios with a limited number of evaluations.

To further analyze the impact of surrogate-assisted evaluation on computational cost, this paper records the total runtime required for each algorithm to complete 50 generations of evolution and conducts 10 independent runs to compare the actual time cost of different algorithms after the reduction in real simulation evaluations.

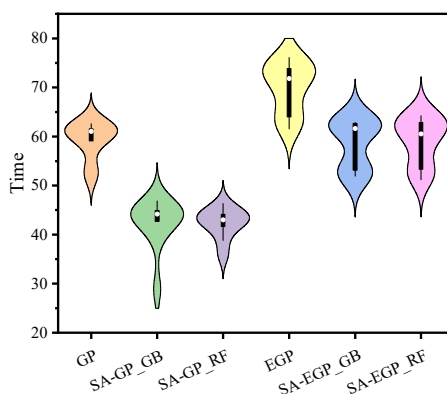


Figure 15. Runtime comparison of different algorithms for 50 generations.

Table 16. Average optimization performance and computational efficiency of different algorithms over 10 independent runs.

Algorithm	HV	IGD	Real evaluations	Surrogate evaluations	Total time
GP	0.971013112	0.014343432	13305	0	59.370213056
SA-GP_GB	0.953512257	0.019034076	5289.1	7668	42.372601085
SA-GP_RF	0.944804215	0.021311760	5245	7618	42.344077732
EGP	0.978709408	0.015224628	12600	0	69.976057413
SA-EGP_GB	0.978839883	0.014141102	8282	4984	58.490403104
SA-EGP_RF	0.972919205	0.019514443	8136	4901	59.085439401

As shown in Figure 15 and Table 16, compared with the baseline algorithms, the runtime of SA-GP_GB and SA-GP_RF is significantly shortened after the introduction of the surrogate model, and the time distributions of the two surrogate models are relatively close. For EGP, SA-EGP_GB and SA-EGP_RF also show a certain time advantage, but the overall improvement is relatively limited. Overall, the surrogate-assisted method can reduce the computational overhead of the algorithm by reducing the number of real simulation evaluations. This further demonstrates that the surrogate model can replace part of the real simulation evaluation at a lower computational cost, thereby reducing the calls to the high-cost simulation environment during evolution. At the same time, the surrogate-assisted

mechanism also introduces additional overhead such as feature computation, model training, prediction, and candidate individual screening, so the reduction in real evaluation times does not translate proportionally into a decrease in total runtime. For EGP, because an ensemble individual contains multiple dispatching rules, its evaluation process itself has higher computational complexity, so the time improvement brought by the surrogate model is relatively limited. In summary, surrogate-assisted evaluation can effectively reduce the computational burden caused by real simulation and improve the overall computational efficiency of the algorithm while maintaining search performance.

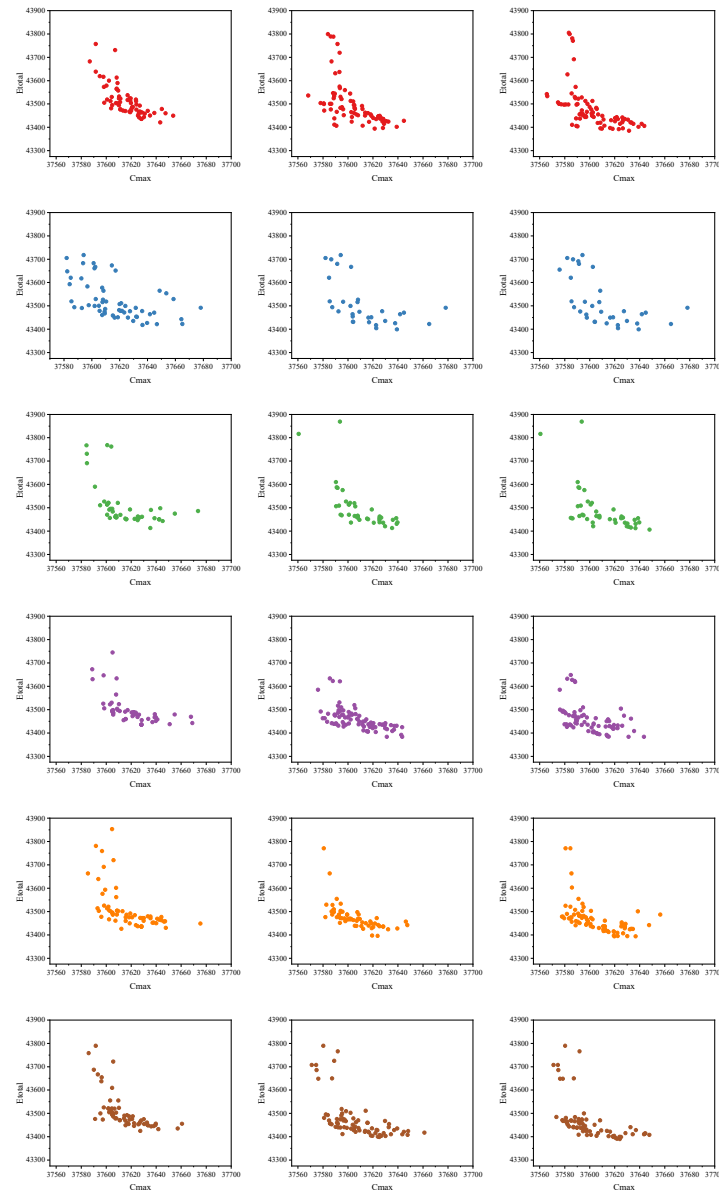


Figure 16. Scatterplots of Pareto front solutions for GP, SA-GP_GB, SA-GP_RF, EGP, SA-EGP_GB, and SA-EGP_RF (from top to bottom) under evaluation counts of 2500, 10,000, and 20,000 (from left to right).

Figure 16 shows the evolution of the Pareto front solutions for the six algorithms under different numbers of evaluations. From the scatterplots, it is clear that as the number of evaluations increases, all algorithms except SA-GP_GB and SA-GP_RF exhibit a clear convergence trend. Meanwhile, at 2500 evaluations, the algorithms incorporating surrogate models demonstrate better convergence performance than the baseline algorithms GP and EGP. However, as the number of evaluations increases, the convergence of SA-EGP_GB and SA-EGP_RF gradually becomes comparable to that of EGP, while still maintaining more excellent population individuals.

For SA-GP_GB and SA-GP_RF, their Pareto solutions further illustrate the deficiencies observed in the convergence curves. Although these two methods can discover local extreme solutions at certain stages, such as solutions with low makespan (Cmax) or low energy consumption, the overall continuity and distribution uniformity of their Pareto fronts are insufficient, making it difficult to form a stable and complete Pareto front. In particular, SA-GP_RF has relatively few valid solution points at 10,000 and 20,000 evaluations, suggesting that its search process may fall into a local optimum early on, resulting in insufficient population diversity and an inability to fully explore the trade-off between makespan and energy consumption.

Overall, the introduction of surrogate models primarily enhances the exploration speed and coverage of the objective space in the early stage. Algorithms equipped with surrogate models can identify the rough shape of the Pareto front earlier. However, as the number of evaluations increases, the original EGP and GP, relying on true objective function evaluations, gradually correct their search directions and ultimately obtain high-quality, stable Pareto solution sets in the final stage. Therefore, SA-EGP_RF and SA-EGP_GB can obtain non-dominated scheduling solutions with good objective space coverage under a limited real simulation evaluation budget.

7. Conclusion

To address the problem of prohibitively high simulation evaluation costs in ensemble genetic programming, this paper proposes a surrogate-assisted evaluation framework tailored to ensembles for dynamic multi-objective job shop scheduling. Within this framework, a unified phenotypic feature representation method is introduced, which maps both single dispatching rules and ensemble individuals into the same behavioral feature space, thereby enabling unified surrogate modeling and coevolution of an individual-ensemble hybrid population.

For ensemble individuals, a structure-aware feature modeling strategy is further introduced. By integrating behavioral features, diversity measures, and decision conflict information, a unified ensemble feature representation is formed. Based on this feature space, separate surrogate models are constructed for each optimization objective to efficiently predict the performance of candidate solutions. On this basis, a hierarchical surrogate-assisted evaluation mechanism is designed: candidate solutions are first evaluated via the surrogate models to obtain multi-objective predictions, and non-dominated sorting is performed based on the predicted results. Subsequently, only a subset of solutions that perform well in the sorting undergoes actual simulation evaluation, while the remaining solutions participate in the evolutionary process using their predicted values, thereby significantly reducing computational cost while maintaining solution quality.

This method has both theoretical and practical significance. From a theoretical perspective, it demonstrates how to extend surrogate models to ensemble individuals in order to handle their coupled behaviors and interactions. In the dynamic job shop scenarios considered in the experiments, the

method significantly reduces simulation-based computational cost while maintaining high solution quality, indicating its potential for further extension to larger-scale and more complex dynamic scheduling scenarios. However, the validation in this paper is still mainly based on a simulation environment; therefore, its practical industrial applicability and large-scale scalability still need to be further verified through a wider range of scenarios and real industrial cases.

Although this paper employs the evaluation ratio p to dynamically adjust the evaluation ratio of the surrogate model, this remains a linear adjustment method, which cannot respond to changing shop state in a timely manner when facing dynamic job shops. Therefore, future research directions should include extending this linear adjustment to online adaptive surrogate modeling, incorporating uncertainty-aware prediction, and integrating reinforcement learning agents to further enhance decision-making capabilities in dynamic and uncertain scheduling scenarios. These extensions are expected to improve the adaptability and robustness of ensemble evolutionary methods in complex industrial environments.

Author contributions

Gang Wang: Conceptualization, Methodology, Software, Formal analysis, Investigation, Data curation, Validation, Visualization, Writing – original draft. Adilanmu Sitahong: Supervision, Methodology, Writing – review & editing. Yiping Yuan: Validation, Writing – review & editing. Peiyin Mo: Writing – review & editing. Areziguli Wubuli: Supervision, Writing – review & editing. Junyan Ma: Writing – review & editing.

Use of Generative-AI tools declaration

During the preparation of this work, no generative AI tools were used. The authors used Grammarly only for language polishing, including grammar and spelling checks. The authors reviewed and edited all content and take full responsibility for the content of the publication.

Acknowledgments

This research was funded by the National Natural Science Foundation of China (No. 72661036); the Youth Science Fund of the Natural Science Foundation of Xinjiang Uygur Autonomous Region (No. 2023D01C177); Key Technology Research and Achievement Transformation of Intelligent Manufacturing Execution System for High end Agricultural Machinery Equipment (No. 2025B04005-0001); Key Technology Research and Development of Large scale Low cost Magnesium Electrolytic Cell (No. HGX2025KJXM008), and Xinjiang Uygur Autonomous Region Key R&D Special Program—Department-Local Joint Initiative (No. 2024B04003-1).

Conflict of interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Y. Feng, Q. Wang, Y. Gao, J. Cheng, J. Tan, Energy-efficient job-shop dynamic scheduling system based on the cyber-physical energy-monitoring system, *IEEE Access*, **6** (2018), 52238–52247. <https://doi.org/10.1109/ACCESS.2018.2869048>
2. L. Ge, C. Su, Q. Cheng, H. Zheng, Z. Liu, Y. Guo, et al., Impacts of global manufacturing transfer on energy intensity, consumption, and transition: an industry linkage perspective, *Habitat Int.*, **161** (2025), 103410. <https://doi.org/10.1016/j.habitatint.2025.103410>
3. K. Lo, H. Li, M. Wang, Energy conservation in China's energy-intensive enterprises: an empirical study of the ten-thousand enterprises program, *Energy Sustain. Dev.*, **27** (2015), 105–111. <https://doi.org/10.1016/j.esd.2015.05.002>
4. A. Hasanbeigi, L. Price, A review of energy use and energy efficiency technologies for the textile industry, *Renew. Sustain. Energy Rev.*, **16** (2012), 3648–3665. <https://doi.org/10.1016/j.rser.2012.03.029>
5. L. He, R. Chiong, W. Li, S. Dhakal, Y. Cao, Y. Zhang, Multiobjective optimization of energy-efficient JOB-shop scheduling with dynamic reference point-based fuzzy relative entropy, *IEEE Trans. Ind. Inform.*, **18** (2022), 600–610. <https://doi.org/10.1109/TII.2021.3056425>
6. J. Duan, J. Wang, Energy-efficient scheduling for a flexible job shop with machine breakdowns considering machine idle time arrangement and machine speed level selection, *Comput. Ind. Eng.*, **161** (2021), 107677. <https://doi.org/10.1016/j.cie.2021.107677>
7. R. Li, W. Gong, C. Lu, L. Wang, A learning-based memetic algorithm for energy-efficient flexible job-shop scheduling with type-2 fuzzy processing time, *IEEE Trans. Evol. Comput.*, **27** (2023), 610–620. <https://doi.org/10.1109/TEVC.2022.3175832>
8. K. Akram, M. U. Bhutta, S. I. Butt, S. I. Husain Jaffery, M. Khan, A. Zeb, et al., A Pareto-optimality based black widow spider algorithm for energy efficient flexible job shop scheduling problem considering new job insertion, *Appl. Soft Comput.*, **164** (2024), 111937. <https://doi.org/10.1016/j.asoc.2024.111937>
9. G. May, B. Stahl, M. Taisch, V. Prabhu, Multi-objective genetic algorithm for energy-efficient job shop scheduling, *Int. J. Prod. Res.*, **53** (2015), 7071–7089. <https://doi.org/10.1080/00207543.2015.1005248>
10. T. Jiang, C. Zhang, H. Zhu, J. Gu, G. Deng, Energy-efficient scheduling for a job shop using an improved whale optimization algorithm, *Mathematics*, **6** (2018), 220. <https://doi.org/10.3390/math6110220>
11. M. Xu, Y. Mei, F. Zhang, M. Zhang, Learn to optimise for job shop scheduling: a survey with comparison between genetic programming and reinforcement learning, *Artif. Intell. Rev.*, **58** (2025), 160. <https://doi.org/10.1007/s10462-024-11059-9>
12. Y. Zeiträg, J. R. Figueira, N. Horta, R. Neves, Surrogate-assisted automatic evolving of dispatching rules for multi-objective dynamic job shop scheduling using genetic programming, *Expert Syst. Appl.*, **209** (2022), 118194. <https://doi.org/10.1016/j.eswa.2022.118194>
13. S. Nguyen, M. Zhang, M. Johnston, K. C. Tan, Learning iterative dispatching rules for job shop scheduling with genetic programming, *Int. J. Adv. Manuf. Technol.*, **67** (2013), 85–100. <https://doi.org/10.1007/s00170-013-4756-9>
14. S. Nguyen, M. Zhang, M. Johnston, K. C. Tan, Genetic programming for evolving due-date assignment models in job shop environments, *Evol. Comput.*, **22** (2014), 105–138. https://doi.org/10.1162/EVCO_a_00105

15. F. Zhang, Y. Mei, S. Nguyen, M. Zhang, Collaborative multifidelity-based surrogate models for genetic programming in dynamic flexible job shop scheduling, *IEEE Trans. Cybern.*, **52** (2022), 8142–8156. <https://doi.org/10.1109/TCYB.2021.3050141>
16. T. Soule, J. A. Foster, Effects of code growth and parsimony pressure on populations in genetic programming, *Evol. Comput.*, **6** (1998), 293–309. <https://doi.org/10.1162/evco.1998.6.4.293>
17. Z. Sheikh Khozani, M. J. S. Safari, A. Danandeh Mehr, W. H. Melini Wan Mohtar, An ensemble genetic programming approach to develop incipient sediment motion models in rectangular channels, *J. Hydrol.*, **584** (2020), 124753. <https://doi.org/10.1016/j.jhydrol.2020.124753>
18. M. Xu, Y. Mei, F. Zhang, M. Zhang, Genetic Programming for Dynamic Flexible Job Shop Scheduling: Evolution With Single Individuals and Ensembles, *IEEE Trans. Evol. Comput.*, **28** (2024), 1761–1775. <https://doi.org/10.1109/TEVC.2023.3334626>
19. A. L. V. Coelho, E. Fernandes, K. Faceli, Inducing multi-objective clustering ensembles with genetic programming, *Neurocomputing*, **74** (2010), 494–498. <https://doi.org/10.1016/j.neucom.2010.09.014>
20. Y. Jin, Surrogate-assisted evolutionary computation: recent advances and future challenges, *Swarm Evol. Comput.*, **1** (2011), 61–70. <https://doi.org/10.1016/j.swevo.2011.05.001>
21. Y. Zhang, S. Bhattacharyya, Genetic programming in classifying large-scale data: an ensemble method, *Inf. Sci.*, **163** (2004), 85–101. <https://doi.org/10.1016/j.ins.2003.03.028>
22. M. Makhoabenyane, S. Guo, E. Leburu, Towards Sustainable Factories: A Systematic Review of Energy-Conscious Job-Shop Scheduling Models and Algorithms, *Sustainability*, **17** (2025), 11330. <https://doi.org/10.3390/su172411330>
23. S. Zhang, J. Zhong, H. Yang, Z. Li, G. Liu, A study on PGEP to evolve heuristic rules for FJSSP considering the total cost of energy consumption and weighted tardiness, *Comput. Appl. Math.*, **38** (2019), 185. <https://doi.org/10.1007/s40314-019-0934-1>
24. S. Yang, L. Meng, S. Ullah, B. Zhang, H. Sang, P. Duan, MILP modeling and optimization of multi-objective three-stage flexible job shop scheduling problem with assembly and AGV transportation, *IEEE Access*, **13** (2025), 25369–25386. <https://doi.org/10.1109/ACCESS.2025.3535825>
25. R. Wang, Y. Jing, C. Gu, S. He, J. Chen, End-to-end multitarget flexible job shop scheduling with deep reinforcement learning, *IEEE Internet Things J.*, **12** (2025), 4420–4434. <https://doi.org/10.1109/JIOT.2024.3485748>
26. Y. Tian, Z. Gao, L. Zhang, Y. Chen, T. Wang, A multi-objective optimization method for flexible job shop scheduling considering cutting-tool degradation with energy-saving measures, *Mathematics*, **11** (2023), 324. <https://doi.org/10.3390/math11020324>
27. J. M. Kim, K. Han, B. S. Choi, S. H. Song, M. Park, M. H. Choi, Effect of coal-fired power plant flexible operating method on boiler header fatigue life, *J. Mech. Sci. Technol.*, **38** (2024), 5113–5121. <https://doi.org/10.1007/s12206-024-0844-z>
28. A. M. Sahlodin, P. I. Barton, Optimal Campaign Continuous Manufacturing, *Ind. Eng. Chem. Res.*, **54** (2015), 11344–11359. <https://doi.org/10.1021/acs.iecr.5b01376>
29. E. Xu, Y. Li, Y. Liu, J. Du, X. Gao, Energy saving scheduling strategy for job shop under TOU and tiered electricity price, *Alex. Eng. J.*, **61** (2022), 459–467. <https://doi.org/10.1016/j.aej.2021.06.008>
30. F. Dunke, S. Nickel, Approximate and exact approaches to energy-aware job shop scheduling with dynamic energy tariffs and power purchase agreements, *Appl. Energy*, **380** (2025), 125065. <https://doi.org/10.1016/j.apenergy.2024.125065>

31. H. Qu, X. Tong, M. Cai, Y. Shi, X. Lan, Energy-saving scheduling strategy for variable-speed flexible job-shop problem considering operation-dependent energy consumption, *Expert Syst. Appl.*, **256** (2024), 124952. <https://doi.org/10.1016/j.eswa.2024.124952>
32. W. Shao, Z. Shao, D. Pi, A multi-neighborhood-based multi-objective memetic algorithm for the energy-efficient distributed flexible flow shop scheduling problem, *Neural Comput. Appl.*, **34** (2022), 22303–22330. <https://doi.org/10.1007/s00521-022-07714-3>
33. J. Duan, F. Liu, Q. Zhang, J. Qin, Y. Zhou, Genetic programming hyper-heuristic-based solution for dynamic energy-efficient scheduling of hybrid flow shop scheduling with machine breakdowns and random job arrivals, *Expert Syst. Appl.*, **254** (2024), 124375. <https://doi.org/10.1016/j.eswa.2024.124375>
34. Z. Hou, K. Wang, H. Zou, J. Guo, Y. Li, Energy-efficient flexible job-shop scheduling with multi-cavity imbalances and machine breakdowns using improved multi-objective genetic programming, *Eng. Optim.*, **58** (2026), 2248–2275. <https://doi.org/10.1080/0305215X.2025.2559840>
35. A. Sitahong, Y. Yuan, M. Li, J. Ma, Z. Ba, Y. Lu, Learning dispatching rules via novel genetic programming with feature selection in energy-aware dynamic job-shop scheduling, *Sci. Rep.*, **13** (2023), 8558. <https://doi.org/10.1038/s41598-023-34951-w>
36. I. D. Mienye, Y. Sun, A survey of ensemble learning: concepts, algorithms, applications, and prospects, *IEEE Access*, **10** (2022), 99129–99149. <https://doi.org/10.1109/ACCESS.2022.3207287>
37. A. Mohammed, R. Kora, A comprehensive review on ensemble deep learning: opportunities and challenges, *J. King Saud Univ. Comput. Inf. Sci.*, **35** (2023), 757–774. <https://doi.org/10.1016/j.jksuci.2023.01.014>
38. M. Đurasević, D. Jakobović, Creating dispatching rules by simple ensemble combination, *J. Heuristics*, **25** (2019), 959–1013. <https://doi.org/10.1007/s10732-019-09416-x>
39. E. Hart, K. Sim, A hyper-heuristic ensemble method for static job-shop scheduling, *Evol. Comput.*, **24** (2016), 609–635. https://doi.org/10.1162/EVCO_a_00183
40. J. Park, Y. Mei, S. Nguyen, G. Chen, M. Zhang, An investigation of ensemble combination schemes for genetic programming based hyper-heuristic approaches to dynamic job shop scheduling, *Appl. Soft Comput.*, **63** (2018), 72–86. <https://doi.org/10.1016/j.asoc.2017.11.020>
41. J. Park, S. Nguyen, M. Zhang, M. Johnston, Evolving ensembles of dispatching rules using genetic programming for job shop scheduling, *Genet. Program.*, (2015), 92–104. https://doi.org/10.1007/978-3-319-16501-1_8
42. Z. Huang, Y. Mei, F. Zhang, M. Zhang, Multitask linear genetic programming with shared individuals and its application to dynamic job shop scheduling, *IEEE Trans. Evol. Comput.*, **28** (2024), 1546–1560. <https://doi.org/10.1109/TEVC.2023.3263871>
43. Y. Mei, S. Nguyen, B. Xue, M. Zhang, An efficient feature selection algorithm for evolving job shop scheduling rules with genetic programming, *IEEE Trans. Emerg. Top. Comput. Intell.*, **1** (2017), 339–353. <https://doi.org/10.1109/TETCI.2017.2743758>
44. Y. Zhou, J. Yang, Z. Huang, Automatic design of scheduling policies for dynamic flexible job shop scheduling via surrogate-assisted cooperative co-evolution genetic programming, *Int. J. Prod. Res.*, **58** (2020), 2561–2580. <https://doi.org/10.1080/00207543.2019.1620362>
45. S. Nguyen, M. Zhang, K. C. Tan, Surrogate-assisted genetic programming with simplified models for automated design of dispatching rules, *IEEE Trans. Cybern.*, **47** (2017), 2951–2965. <https://doi.org/10.1109/TCYB.2016.2562674>

46. T. Hildebrandt, J. Branke, On using surrogates with genetic programming, *Evol. Comput.*, **23** (2015), 343–367. https://doi.org/10.1162/EVCO_a_00133
47. S. Nguyen, M. Zhang, D. Alahakoon, K. C. Tan, People-centric evolutionary system for dynamic production scheduling, *IEEE Trans. Cybern.*, **51** (2021), 1403–1416. <https://doi.org/10.1109/TCYB.2019.2936001>
48. F. Zhang, Y. Mei, S. Nguyen, K. C. Tan, M. Zhang, Instance-rotation-based surrogate in genetic programming with brood recombination for dynamic job-shop scheduling, *IEEE Trans. Evol. Comput.*, **27** (2023), 1192–1206. <https://doi.org/10.1109/TEVC.2022.3180693>
49. T. G. Dietterich, Ensemble Methods in Machine Learning, *Mult. Classif. Syst.*, (2000), 1–15. https://doi.org/10.1007/3-540-45014-9_1
50. Z. Wang, J. Zhang, S. Yang, An improved particle swarm optimization algorithm for dynamic job shop scheduling problems with random job arrivals, *Swarm Evol. Comput.*, **51** (2019), 100594. <https://doi.org/10.1016/j.swevo.2019.100594>
51. G. Varoquaux, L. Buitinck, G. Louppe, O. Grisel, F. Pedregosa, A. Mueller, Scikit-learn: machine learning without learning the machinery, *GetMobile Mob. Comput. Commun.*, **19** (2015), 29–33. <https://doi.org/10.1145/2786984.2786995>



AIMS Press

© 2026 the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (<https://creativecommons.org/licenses/by/4.0>)