*mathematics*

*Article*

# Co-Evolutionary Algorithm for Two-Stage Hybrid Flow Shop Scheduling Problem with Suspension Shifts

**Zhijie Huang, Lin Huang and Debiao Li ***

Management Science and Engineering Department, Fuzhou University, Fuzhou 350116, China; 072103408@fzu.edu.cn (Z.H.); huangl229@foxmail.com (L.H.)
* Correspondence: debiaoli@fzu.edu.cn

**Abstract:** Demand fluctuates in actual production. When manufacturers face demand under their maximum capacity, suspension shifts are crucial for cost reduction and on-time delivery. In this case, suspension shifts are needed to minimize idle time and prevent inventory buildup. Thus, it is essential to integrate suspension shifts with scheduling under an uncertain production environment. This paper addresses the two-stage hybrid flow shop scheduling problem (THFSP) with suspension shifts under uncertain processing times, aiming to minimize the weighted sum of earliness and tardiness. We develop a stochastic integer programming model and validate it using the Gurobi solver. Additionally, we propose a dual-space co-evolutionary biased random key genetic algorithm (DCE-BRKGA) with parallel evolution of solutions and scenarios. Considering decision-makers' risk preferences, we use both average and pessimistic criteria for fitness evaluation, generating two types of solutions and scenario populations. Testing with 28 datasets, we use the value of the stochastic solution (VSS) and the expected value of perfect information (EVPI) to quantify benefits. Compared to the average scenario, the VSS shows that the proposed algorithm achieves additional value gains of 0.9% to 69.9%. Furthermore, the EVPI indicates that after eliminating uncertainty, the algorithm yields potential improvements of 2.4% to 20.3%. These findings indicate that DCE-BRKGA effectively supports varying decision-making risk preferences, providing robust solutions even without known processing time distributions.

**Keywords:** scheduling; suspension shifts; uncertain processing times; biased random key genetic algorithm; co-evolutionary

**MSC:** 68U01

**Citation:** Huang, Z.; Huang, L.; Li, D. Co-Evolutionary Algorithm for Two-Stage Hybrid Flow Shop Scheduling Problem with Suspension Shifts. *Mathematics* **2024**, *12*, 2575. https://doi.org/10.3390/math12162575

Academic Editor: Frank Werner

Received: 23 July 2024
Revised: 16 August 2024
Accepted: 19 August 2024
Published: 20 August 2024

## 1. Introduction

Suspension shifts are crucial for enterprises to adapt to market demand changes and to regulate production capacity by controlling working hours, ensuring on-time order delivery. In actual production, demand fluctuates. When demand is below maximum capacity, suspension shifts reduce production capacity, minimizing machine idle time, operational costs, and inventory due to early completions. Considering that in a production workshop, most processes involve human–machine collaboration and are influenced by fixed work schedules and shift handovers, the decision to suspend operations must be made according to the shift. Additionally, suspending all machines in the workshop simultaneously can maximize cost reduction, so all operations in the same workshop should be suspended according to the same shift. Therefore, production planning typically schedules workshop suspension shifts based on actual orders, creating non-production periods that constrain workshop scheduling.

However, if the suspension shift plan is not effectively integrated with production scheduling, it can fail to reduce operational costs and inventory, leading to lower fulfillment rates and financial losses. For example, as shown in Figure 1, there are six shifts, and the release times and due dates of three jobs, $J_1$, $J_2$, and $J_3$, are $r_1$, $r_2$, and $r_3$, and $d_1$, $d_2$, and $d_3$, respectively. In case 1, job $J_2$'s arrival time $r_2$ causes machine idling, which the suspension shift plan cannot predict, leading to early completion, excess operational costs, and increased inventory. Even if the operation is suspended in shift 2, early completion and inventory issues persist. In case 2, an unreasonable suspension shift plan delays job $J_3$, resulting in penalties and customer loss. In case 3, a reasonable suspension shift plan effectively reduces operational costs and inventory. Thus, it is essential to integrate workshop suspension shifts with production scheduling. This ensures that the suspension shift plan effectively regulates production capacity and promotes on-time delivery.

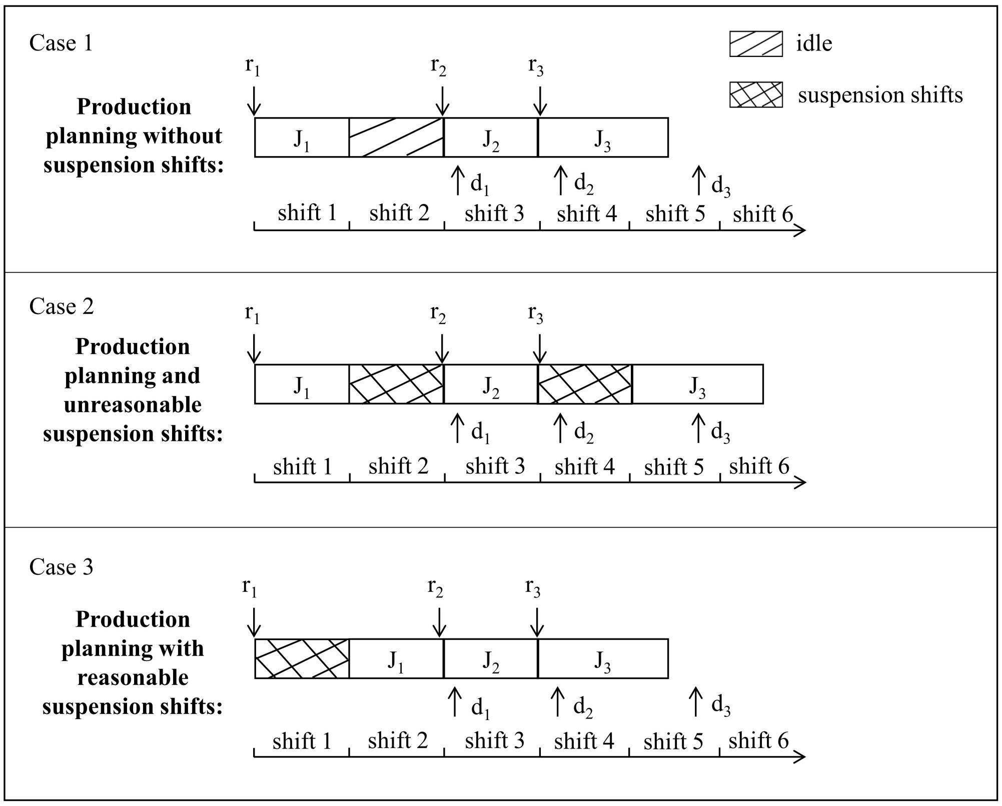

Case 1
idle
suspension shifts
Production planning without suspension shifts:
r1
r2
r3
J1
J2
J3
d1
d2
d3
shift 1
shift 2
shift 3
shift 4
shift 5
shift 6
Case 2
Production planning and unreasonable suspension shifts:
Case 3
Production planning with reasonable suspension shifts:

**Figure 1.** Co-optimization benefits analysis.

The hybrid flow shop scheduling problem (HFSP) is a class of NP-hard combinatorial optimization problems [1] that is widely applied in industries such as chemicals, semiconductors, metallurgy, textiles, and logistics. The HFSP involves multiple production stages with parallel machines, where jobs must be processed in sequence. This paper investigates the two-stage hybrid flow shop scheduling problem (THFSP), which is a common variant of HFSP that is distinguished by its restriction to two production stages, which simplifies the general problem while retaining significant theoretical and practical importance [2]. The THFSP is particularly relevant in applications such as PCB assembly, metal fabrication, and two-stage packaging processes. Meanwhile, the heterogeneity of production and the diversity of orders make it difficult to accurately predict the processing time of jobs. This makes it even harder to ensure the rationality of pre-arranged workshop suspension shifts. If the actual processing time exceeds the estimated time, work orders will be overdue. Conversely, if the actual processing time is shorter than the estimated time, work orders will be completed early. Therefore, researching two-stage hybrid flow shop scheduling with suspension shifts is of significant importance in an environment with uncertain processing times.

The goal of this work is to provide decision-makers with robust scheduling solutions in the workshop. Considering suspension shifts and uncertain processing times, we develop a method based on a dual-space co-evolutionary biased random key genetic algorithm (DCE-BRKGA). In this method, parallel populations of solutions and scenarios co-evolve, with each relying on the other for the fitness evaluation of their individuals [3]. This approach aims to generate representative and diverse scenario populations and measure their impact on solution populations. Simultaneously, the solutions evolve based on different decision-making risk profiles, and we collectively evaluate the performance of the scenarios.

The primary contributions of this paper are related to the proposed problems, solution methods, and comprehensive experiments:

- We propose a stochastic integer programming model to address the two-stage hybrid flow shop scheduling problem with suspension shifts under uncertain processing times. This model regards suspension shifts as a crucial tool to adjust capacity and ensure on-time delivery, and it aims to minimize the weighted sum of job earliness and tardiness. By incorporating constraints on job sequencing, machine utilization, and suspension shifts, the model can flexibly adjust capacity in complex production environments, ensuring the effective execution of production plans.
- In a deterministic environment, we test the performance of a biased random key genetic algorithm (BRKGA) by comparing it with Gurobi and the random key genetic algorithm (RKGA). The BRKGA's final solutions are, on average, 7.08% worse than Gurobi's optimal solutions. Additionally, we compare the optimal solutions generated by RKGA and BRKGA under two different fitness criteria. The results show that BRKGA's optimal solutions are, on average, 49.7% and 50.4% better than those of RKGA, demonstrating its ability to produce superior solutions.
- For uncertain environments, we propose the DCE-BRKGA, which incorporates parallel population co-evolution. When applying the pessimistic criterion, we designed a specialized scenario fitness function based on Lemmas 1 and 2. It transforms scenario evolution into a maximization problem, while solution evolution is treated as a minimization problem. It clarifies the search direction and speeds up the convergence of the algorithm.
- We evaluate the performance of the DCE-BRKGA by the value of the stochastic solution (VSS) and the expected value of perfect information (EVPI). The VSS shows that the proposed algorithm can achieve additional value gains of 0.9% to 69.9% compared to the average scenario. Furthermore, the EVPI indicates that after eliminating uncertainty, the algorithm yields potential improvements of 2.4% to 20.3%. This is particularly useful for manufacturers as it helps them understand the value of investing in solutions to reduce uncertainty or to improve prediction accuracy.

The structure of this article is as follows. Initially, the proposed problem is stated and a mathematical model is presented in Section 3. Subsequently, a co-evolutionary algorithm is introduced to solve the NP-hard problem in Section 4, and the results of computational tests are discussed in Section 5. Finally, conclusions are drawn and future work and research directions are discussed in Section 6.

## 2. Literature Review

This study employs a co-evolutionary algorithm to conduct two-stage hybrid flow shop scheduling with suspension shifts in an environment of uncertain processing times. The research expands upon previous works in three key aspects: workshop scheduling considering suspending operations, workshop scheduling with uncertain processing times, and the application of co-evolutionary algorithms.

### *2.1. Workshop Scheduling Research Considering Suspending Operations*

Workshop suspensions adjust production capacity by controlling machine availability time. Research on workshop scheduling considering machine availability can be categorized into two types:

Optimizing the scheduling plan: This approach treats machine availability as a constraint. For instance, Lin et al. [4] considered constraints due to regular maintenance and shift changes, employing an online algorithm to address the parallel machine scheduling problem for uninterrupted job processing. Nguyen [5] introduced a polynomial-time approximation scheme for minimizing the total completion time of two parallel machines. Yu et al. [6] examined the proportional flow shop scheduling problem with periodic machine maintenance. Detti et al. [7] aimed to minimize completion time in single-machine scheduling problems with uncertain maintenance activities using robust scheduling methods.

Collaborative optimization research: This type focuses on the synergy between machine availability planning and scheduling planning. Lee and Leon [8] studied the collaborative optimization of single-machine variable-rate scheduling, regarding maintenance as a tool to enhance equipment utilization and discussing strategies to minimize total processing time and delays within a single maintenance cycle. Nourelfath et al. [9] proposed a joint model for parallel machine system scheduling and maintenance. Lu et al. [10] suggested a new integrated model to dynamically adjust maintenance plans, noting that periodic maintenance might lead to excessive maintenance. Liu Yu et al. [11] introduced a parallel-machine production synchronization evaluation model, integrating production plans with preventive maintenance scheduling using the moment-matching method. Zheng et al. [12] explored the energy-saving two-stage hybrid flow shop scheduling problem, combining time-of-use electricity pricing decisions with machine working states and production plans to optimize the maximum completion time and total energy consumption.

These studies underscore the significance of joint decision-making in machine availability planning and production planning for enhancing production efficiency, reducing costs, and ensuring product quality. This area has garnered substantial attention in the academic community.

### 2.2. *Workshop Scheduling Research Considering Uncertain Processing Time*

Early research on workshop scheduling optimization was often based on idealized assumptions. However, uncertainties in actual production environments—such as fluctuations in equipment performance, failures, and maintenance needs—make accurate prediction of processing times challenging. This makes the in-depth study of processing time uncertainty crucial for optimizing scheduling, improving efficiency, and reducing risks. Current research on scheduling under uncertain processing times can be divided into three main directions:

Rescheduling: This approach generates an initial scheduling plan based on current workshop information and adjusts the plan according to actual conditions when random disturbances occur. For example, Zadeh [13] initially schedules based on estimated processing times and then reschedules after determining the actual times, using the artificial bee colony algorithm to solve the dynamic flexible workshop scheduling problem with the goal of minimizing the maximum completion time. Framinan et al. [14] explored using real-time completion times of pipeline operations to reschedule jobs, demonstrating through computational experiments that rescheduling policies are effective only if the processing time variability is low and the initial plan quality is good.

Stochastic scheduling: This approach uses known probability distributions or relevant empirical data about job processing times, employing methods based on expected indicators for decision-making. Yue et al. [15] addressed the timing window allocation scheduling problem with stochastic processing times, using a branch-and-bound algorithm to solve the single-machine scheduling problem optimally. Ghaedy-Heidary et al. [16] proposed a simulation optimization framework combining genetic algorithms and simulation models for the stochastic flexible job shop scheduling problem. Liu et al. [17] studied the stochastic parallel-machine scheduling problem with uncertain job arrivals and processing times, proposing a two-stage method to minimize the total cost: first assigning jobs when uncertainty is unknown, then scheduling when uncertainty is known.

Robust scheduling: This approach pre-identifies and evaluates potential uncertainty events during production, incorporating their impact into the preliminary scheduling plan to minimize worst-case performance. Lu et al. [18] studied the single-machine scheduling problem with uncertain processing times, using simple iterative improvement heuristics and simulated annealing heuristics. Wang et al. [19] investigated the robust scheduling problem for identical parallel machines with uncertain processing times, considering the possibility of outsourcing jobs. Xiao et al. [20] examined the job shop scheduling problem with stochastic deteriorating processing times, measuring schedule robustness by the expected deviation between realized and initial completion times.

Therefore, studying the randomness of processing times is crucial for optimizing scheduling and reducing risks. Scholars have proposed three methods—rescheduling, stochastic scheduling, and robust scheduling—to address this challenge, each with its advantages and disadvantages and suitable for different production environments. In practical applications, it is necessary to choose the appropriate scheduling strategy based on specific conditions.

### *2.3. Application Research on Co-Evolutionary Algorithms*

To address the complex optimization challenges characterized by large scales, multiple objectives, and uncertainty, co-evolutionary algorithms have become a significant research area in evolutionary computation. These algorithms improve performance, efficiency, and robustness [21–23]. Co-evolutionary algorithms have received considerable attention in manufacturing production [24] and various other fields [25]. Population cooperation is the primary method to achieve co-evolution [26]. Co-evolution can be categorized into competitive and cooperative types based on different inter-population evaluation methods. The following discusses research on competitive and cooperative co-evolutionary algorithms for solving uncertainty problems.

Competitive co-evolutionary algorithms are based on the principle of intermediate competition in ecology, where improvement in one population exerts selective pressure on other populations, thereby affecting their evolution. Gu et al. [27] proposed a competitive co-evolutionary quantum genetic algorithm to solve job shop scheduling problems with uncertain processing times. They designed three interspecific competition strategies for population evaluation and dynamically adjusted the population sizes to increase the genetic diversity and prevent premature convergence.

Cooperative co-evolutionary algorithms decompose complex problems into several subproblems, which are solved by various populations through multi-population cooperation for collaborative evaluation. Herrmann [28] was the first to design a dual-population co-evolutionary genetic algorithm for robust scheduling of parallel machines under uncertain processing times. Jensen [29] introduced a ranking-based scenario fitness evaluation, correcting the symmetry and bias issues of the original method. Oliveira et al. [3] proposed a dual-space co-evolutionary biased random key genetic algorithm for car rental problems with uncertain demand, where solution and scenario parallel populations co-evolve.

Current research shows that co-evolutionary algorithms have been widely applied in multiple fields, but there has been less research on dealing with uncertainties. The application of co-evolutionary algorithms to uncertainty problems can be categorized into competitive and cooperative types. Competitive co-evolutionary algorithms improve algorithmic performance through competitive strategies, while cooperative co-evolutionary algorithms decompose uncertainty problems, separating uncertain factors from certain ones to allow parallel populations of solutions and scenarios to co-evolve. Cooperative methods are simple to operate and have strong applicability, supporting the design of scenarios with different decision risk preferences and ensuring solution quality. However, their application has not been fully explored and utilized, necessitating further in-depth research.

## 3. Problem Definition

This paper discusses the two-stage hybrid flow shop scheduling problem with suspension shifts under uncertain processing times, aiming to obtain production schedules and a workshop suspension shift plan during the planning period. The production schedule aims for products to be completed exactly on the due date; early completion increases operational and inventory costs, while late completion results in penalties for breach of contract. Therefore, the objective of this study's production scheduling is to minimize the weighted sum of earliness and tardiness of jobs, achieving on-time delivery to reduce operational and inventory costs and to minimize the penalties due to delays.

The problem under investigation can be characterized as follows: Given a set of jobs $N = \{1, 2, \cdots, n\}$, each job undergoes processing operations in a two-stage hybrid flow shop, where each stage has $m_1$ and $m_2$ identical parallel machines, respectively. Each job requires two operations, with the first operation processed at the first stage and the second operation processed at the second stage, and each operation can be assigned to any machine $k$ within its corresponding stage. Assuming each job's processing at each stage as a single job, the problem assumptions are as follows:

1. All jobs must pass through the stages in the same order, and a job can only proceed to the next stage after completing the current one;
2. A job cannot be processed on different machines simultaneously;
3. Each machine can process only one job at a time without preemption;
4. Parallel machines at each stage have identical technical features, production capacities, and processing speeds;
5. All jobs and machines are available at time 0;
6. Buffer areas between machines are sufficiently large, allowing processed jobs to wait;
7. Job setup times are included in the processing times or are negligible;
8. Machines cannot process jobs during suspension shifts, but if a job is interrupted by a suspension shift, it can resume processing after the suspension shift ends;
9. All machines follow the same suspension shift schedule to reduce operational costs.

The heterogeneity of workshop environments and the diversity of orders make predicting job processing times challenging, leading to uncertainties in actual production environments. Therefore, this study considers the limited known information about job processing times: specifically, the upper and lower bounds for each stage.

Most manufacturing companies operate on a two-shift system, dividing each day into two 12 h shifts (day and night). When planning a suspension shift schedule, it is necessary to schedule the production of current jobs and determine the shifts during the week when suspension should occur. However, to keep the production line running, the shop must ensure at least four working days per week, allowing for no more than six suspension shifts. Since the objective of this study is to minimize the weighted sum of job earliness and tardiness, we aim to maximize the number of workshop suspension shifts without worsening this objective value, achieving capacity optimization and on-time delivery of orders. Let $t^m$ represent the number of suspension shifts required. The problem then transforms into finding the optimal solution for collaborative optimization by starting from $t^m = 0$ and incrementally adding one suspension shift until the objective value worsens or the number of suspension shifts exceeds six ($t^m > 6$), as shown in Figure 2. Additionally, if a day is divided into three, four, or more shifts, the algorithm can still be applied by simply recalculating the number of shifts and adjusting the constraints on $t^m$ according to the specific situation. For example, three shifts means that a shift is 8 h, and $t^m$ can be calculated accordingly.

### *Problem Modeling*

To formalize the problem, we develop a stochastic integer programming model. Following the detailed problem description and the set of assumptions outlined earlier, we introduce pertinent symbols to enhance clarity, as presented in Table 1.

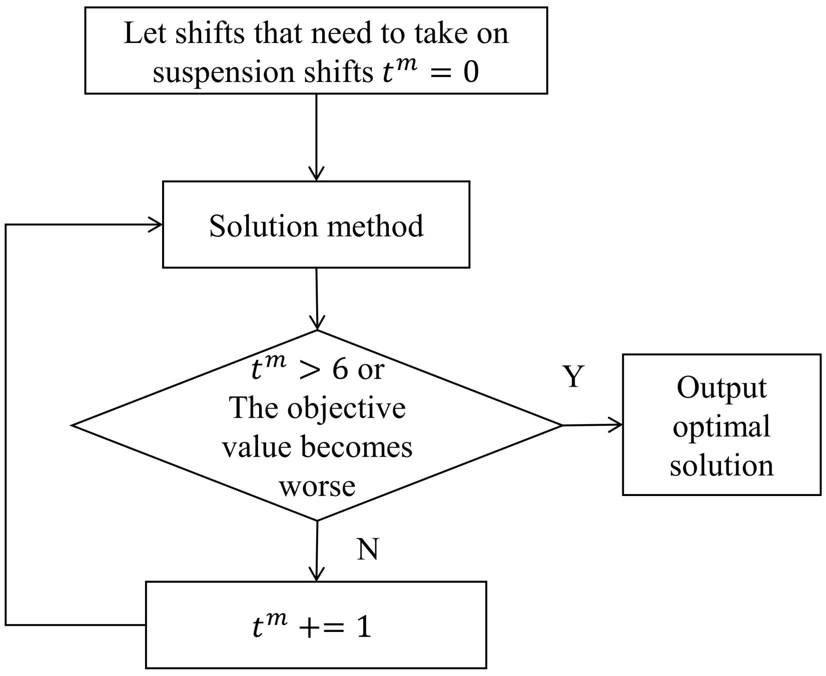

Let shifts that need to take on suspension shifts $t^m = 0$
Solution method
$t^m > 6$ or The objective value becomes worse
Y
Output optimal solution
N
$t^m$ += 1

**Figure 2.** Problem transformation flowchart.

**Table 1.** Mathematical notations.

| Notation | Description |
|---|---|
| $S$ | Stage set $S = \{1, 2\}$ |
| $l$ | Stage symbols, $l \in S$ |
| $M_l$ | A set of machines in stage $l$ |
| $M$ | All machines set |
| $k$ | Machine symbol, $k \in M_l$ |
| $N$ | A set of jobs that does not contain virtual jobs, $N = \{1, 2, \cdots, n\}$ |
| $N^0$ | A set of jobs containing virtual jobs, $N = \{0, 1, 2, \cdots, n\}$, For each stage and machine, introduce a virtual job 0, which precedes the first job on each machine |
| $n$ | Total number of jobs |
| $m_l$ | Total number of machines in stage $l$ |
| $h, i, j$ | Job symbol, $h, i \in N^0, j \in N,\ \ h \neq i \neq j$ |
| $\alpha_i$ | Earliness weight of job $i$ |
| $\beta_i$ | Tardiness weight of job $i$ |
| $\tilde{p}_{il}$ | Stochastic processing time of job $i$ in stage $l$ |
| $d_i$ | Due date of job $i$ |
| $T$ | Shift set, $T = \{1, 2, \cdots, t^0\}$ |
| $t$ | Shift symbol, $t \in T$ |
| $\mu$ | Duration of each shift |
| $t^m$ | Shifts requiring suspensions |
| $s_{il}$ | Start time of job $i$ in stage $l$ |
| $E_i$ | Earliness of job $i$ |
| $T_i$ | Tardiness of job $i$ |
| $M_1$ | Maximum value |

Mathematical Modeling

*Decision variables*:

$Z_{ijlk}$ 0–1 variable: $Z_{ijlk} = 1$ if job $j$ is machined immediately after job $i$ on machine $k$ in stage $l$; otherwise, $Z_{ijlk} = 0$.
$X_t$ 0–1 variable: $X_t = 1$ if a suspension occurs on shift $t$; otherwise, $X_t = 0$.
$Y_{ilt}$ 0–1 variable: $Y_{ilt} = 1$ if the end machining time of job $i$ is at shift $t$ in stage $l$; otherwise, $Y_{ilt} = 0$ .
$R_{ilt}$ 0–1 variable: $R_{ilt} = 1$ if job $i$ is machined across shift $t$ in stage $l$; otherwise, $R_{ilt} = 0$ .
*Optimization model*:

$$Min \sum_{i \in N} (\alpha_i E_i + \beta_i T_i) \tag{1}$$

*Constraints*:

$$E_i \geq 0, \forall i \tag{2}$$

$$E_i - d_i + \left(s_{i2} + \tilde{p}_{i2} + X_t(Y_{i2t} + R_{i2t})\mu\right) \geq 0, \forall i, \forall t \tag{3}$$

$$T_i \geq 0, \forall i \tag{4}$$

$$T_i + d_i - (s_{i2} + \tilde{p}_{i2} + X_t(Y_{i2t} + R_{i2t})\mu) \geq 0, \forall i, \forall t \tag{5}$$

$$\sum_{k \in M_l} \sum_{i \in N^0} Z_{ijlk} = 1, \forall l, \forall j \tag{6}$$

$$\sum_{j \in N} Z_{0jlk} \leq 1, \forall l, \forall k \tag{7}$$

$$Z_{ijlk} + Z_{jilk} \leq 1, \forall k, \forall l, \forall i, \forall j = i+1, \cdots, n \tag{8}$$

$$\sum_{j \in N} Z_{ijlk} - \sum_{h \in N^0} Z_{hilk} \leq 0, \forall l, \forall k, \forall i \tag{9}$$

$$s_{il} + \widetilde{p}_{il} + X_t(Y_{ilt} + R_{ilt})\mu - s_{jl} - M_1(1 - Z_{ijlk}) \leq 0, \forall k, \forall i, \forall j, \forall l, \forall t \tag{10}$$

$$s_{0l} - s_{il} - M_1(1 - Z_{0ilk}) \leq 0, \forall k, \forall i, \forall l \tag{11}$$

$$s_{i1} + \widetilde{P}_{i1} + X_t(Y_{it1} + R_{it1})\mu - s_{i2} \leq 0, \forall i \tag{12}$$

$$s_{il} \geq 0, \forall i, \forall l \tag{13}$$

$$s_{0l} = 0, \forall l \tag{14}$$

$$\sum_{t \in T} X_t - t^m = 0 \tag{15}$$

$$s_{il} + \widetilde{p}_{il} - \mu t - M_1(1 - Y_{ilt}) \leq 0, \forall t, \forall i, \forall l \tag{16}$$

$$s_{il} + \widetilde{p}_{il} - \mu(t-1) + M_1(1 - Y_{ilt}) \geq 0, \forall t, \forall i, \forall l \tag{17}$$

$$s_{il} + \widetilde{p}_{il} - \mu(t+1) - M_1(1 - R_{ilt}) \leq 0, \forall t, \forall i, \forall l \tag{18}$$

$$s_{il} + \widetilde{p}_{il} - \mu t + M_1(1 - R_{ilt}) \geq 0, \forall t, \forall i, \forall l \tag{19}$$

$$Z_{ijlk} \in \{0,1\}, \forall k, \forall i, \forall j, \forall l \tag{20}$$

$$Y_{ilt} \in \{0,1\}, \forall i, \forall l, \forall t \tag{21}$$

$$R_{ilt} \in \{0,1\}, \forall i, \forall l, \forall t \tag{22}$$

Equation (1) describes the objective function of this study as the weighted sum of the earliness and tardiness of jobs. Constraints (2) and (3) define the earliness of jobs, while constraints (4) and (5) define the tardiness of jobs. Constraint (6) ensures that each job is processed by exactly one machine. Constraint (7) ensures that the first job produced cannot exceed the total number of machines available. Constraint (8) ensures the feasibility. Constraint (9) ensures that the immediate precedent and subsequent processes of job j must be on the same machine. Constraints (10) and (11) calculate the start time of each job. If the end time of the previous job falls within a suspension shift or if the job's processing time spans a suspension shift, the start time of the job must include the suspension shift time. Constraint (12) ensures that the start time of the second-stage job is later than the completion time of the same job in the previous stage. Constraint (13) states that all jobs arrive at time 0. Constraint (14) establishes that the earliest production time for each machine is at time 0. Constraint (15) guarantees that the total number of suspension shifts meets the required criterion. Constraints (16) and (17) determine the interval of the shift in which the job ends. Constraints (18) and (19) specify the shifts spanned by the job processing. Finally, constraints (20)–(22) indicate that the decision variables are binary integers.

## 4. Solution Method

As the current problem is an NP-hard problem with uncertain processing times, we adopt the DCE-BRKGA which has the ability to handle uncertainty by co-evolution of the solution populations and scenario populations. This is an innovative extension of the original genetic algorithm, addressing its limitation in handling uncertainty problems. The flowchart of the algorithm is presented in Figure 3.

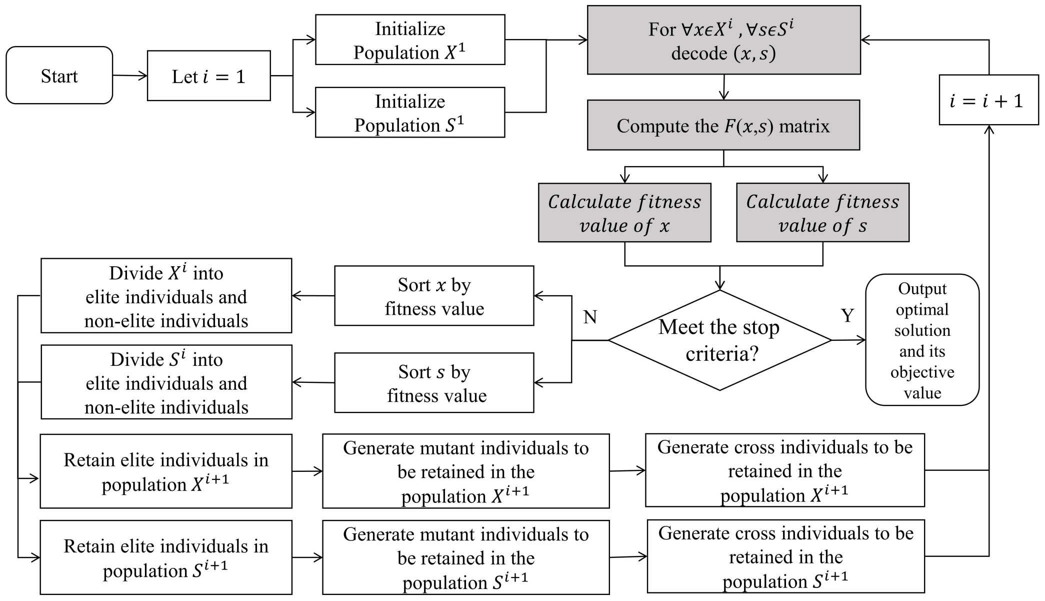

Start
Let $i = 1$
Initialize Population $X^1$
Initialize Population $S^1$
For $\forall x \epsilon X^i$, $\forall s \epsilon S^i$ decode $(x, s)$
Compute the $F(x,s)$ matrix
$i = i + 1$
*Calculate fitness value of x*
*Calculate fitness value of s*
Meet the stop criteria?
N
Y
Output optimal solution and its objective value
Sort $x$ by fitness value
Sort $s$ by fitness value
Divide $X^i$ into elite individuals and non-elite individuals
Divide $S^i$ into elite individuals and non-elite individuals
Retain elite individuals in population $X^{i+1}$
Retain elite individuals in population $S^{i+1}$
Generate mutant individuals to be retained in the population $X^{i+1}$
Generate mutant individuals to be retained in the population $S^{i+1}$
Generate cross individuals to be retained in the population $X^{i+1}$
Generate cross individuals to be retained in the population $S^{i+1}$

**Figure 3.** DCE-BRKGA flowchart.

First, the solution population $X^1$ and scenario population $S^1$ are initialized through random number encoding, with the current iteration number set to $i = 1$, as described in Section 4.1. Each pair of individuals $(x, s)$ formed from the two populations is decoded in Section 4.2 to calculate the objective function of the complete scheduling solution, where $x \in X^i$ and $s \in S^i$. The calculation of the objective function for each pair $(x, s)$ yields the matrix $F(x, s)$. Next, based on the objective function matrix $F(x, s)$, the fitness values of all solutions $x \in X^i$ and scenarios $s \in S^i$ are calculated and sorted according to their fitness values. Considering the differences in risk identification and management capabilities among decision-makers, two types of individual fitness evaluation methods are proposed. Detailed information is introduced in Section 4.3. Finally, all individuals in the population are divided into elite and non-elite individuals. The population is evolved by retaining elite individuals, mutating individuals, and crossing individuals into the next-generation population, as shown in Section 4.4.

### *4.1. Initialization*

#### 4.1.1. Solution Population Initialization

The solution is divided into two parts: a scheduling decision and a workshop suspension shift decision. Given a population size of $P$, the initial population is represented by a $P \times (2n + t)$ matrix $X^i$, where $i$ represents the iteration number, and each row of the matrix represents an individual in the population. For convenience, the part representing the scheduling decisions is denoted by a $P \times 2n$ matrix $A^i$, and the part representing the workshop suspension shift decision is denoted by a $P \times t$ matrix $B^i$.

For the matrix $A^i$, the $j$-th number of the $e$-th row is denoted as $a^i_{ej}$. Thus, the matrix $A^i$ can also be represented by each row $a^i_e$, $A^i = [a^i_1, a^i_2, \cdots, a^i_P]^T$. Each row $a^i_e$ in the matrix contains 2n real numbers, as shown in Figure 4. For convenience, by treating the processing of each stage of $n$ jobs as a separate job, these 2n real numbers represent the scheduling information for 2n jobs. Decoding these 2n real numbers yields the scheduling decision plan. Among them, jobs $j \in \{1, 2, \cdots, n\}$ represent the processing of $n$ jobs in the first stage, and jobs $j \in \{n+1, n+2, \cdots, 2n\}$ represent the processing of $n$ jobs in the second stage. That is, when $j \in \{1, 2, \cdots, n\}$, $a^i_{ej}$ and $a^i_{e,j+n}$ respectively represent the encoding of the first- and second-stage processing information for the same job.

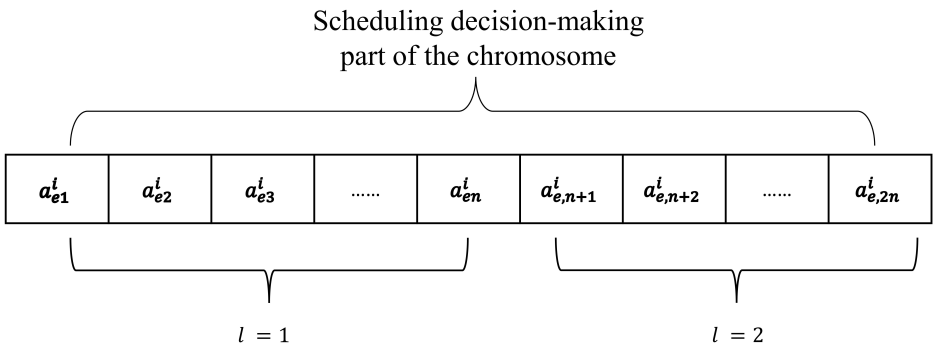


**Figure 4.** Solution chromosomal fragments.

There are $m_1$ machines in the workshop used for processing jobs in the first stage, and there are $m_2$ machines for the second stage. Let $M$ denote the set of all machines in the workshop: $M = \{1, 2, \cdots, m_1 + m_2\}$. The subset $\{1, 2, \cdots, m_1\}$ represents the machines for the first stage, and the subset $\{m_1 + 1, m_1 + 2, \cdots, m_1 + m_2\}$ represents the machines for the second stage. To ensure the feasibility of the scheduling decision $a^i_e$, the encoding method used is shown in Equation (23).

$$a^i_{ej} \in \begin{cases} (0, m_1], & \forall j \in 1, 2, \cdots, n \\ (m_1, m_1 + m_2], & \forall j \in n+1, n+2, \cdots, 2n \end{cases} \tag{23}$$

For the matrix $B^i$, the $j$-th number of the $e$-th row is denoted as $b^i_{ej}$. Thus, the matrix $B^i$ can be represented by each row $b^i_e$, $B^i = [b^i_1, b^i_2, \cdots, b^i_P]^T$. Each row $b^i_e$ in the matrix contains $t$ binary (0–1) variables. Since the number of suspension shifts is $t^m$, to ensure the feasibility of the workshop suspension shift decision $b^i_e$, each row $b^i_e$ needs to satisfy Equation (24).

$$\sum_{j=1}^{t} b^i_{ej} = t^m \tag{24}$$

Let us consider an example: suppose there are three jobs, with two machines in the first stage and three machines in the second stage. The production schedule spans four shifts, with one of these shifts requiring suspension. The population size $P$ is five. The initial population $X^1$ is obtained by combining matrices $A^1$ and $B^1$, as shown below:

$$X^1 = \begin{pmatrix} 0.1136 & 1.1037 & 3.0028 & 4.8345 & 2.6842 & 4.0088 & 0 & 0 & 1 & 0 \\ 0.8455 & 0.7480 & 2.5718 & 2.5088 & 3.9093 & 4.7653 & 1 & 0 & 0 & 0 \\ 1.0174 & 1.4762 & 4.8187 & 3.4526 & 2.9920 & 4.7296 & 0 & 1 & 0 & 0 \\ 1.6155 & 0.2605 & 4.1876 & 4.1897 & 4.9889 & 4.1250 & 1 & 0 & 0 & 0 \\ 0.2479 & 1.5545 & 3.7086 & 3.1028 & 4.0237 & 2.4210 & 0 & 0 & 0 & 1 \end{pmatrix}$$

### 4.1.2. Scenario Population Initialization

The initial population of the scenario is represented by a $P \times 2n$ matrix $S^i$. In matrix $S^i$, the first $n$ numbers in each row represent the processing times of $n$ jobs in the first stage, the $j$-th number of the $e$-th row is denoted as $\tilde{p}^i_{ej1}$, the latter $n$ numbers represent the processing times of $n$ jobs in the second stage, and the $(j+n)$-th number of the $e$-th row is denoted as $\tilde{p}^i_{ej2}$. Due to limited information about the job processing times, it is only known that the processing time of job $j$ in stage $l$ has an upper bound $U_{jl}$ and a lower bound $L_{jl}$. Therefore, the generated job processing times must follow a uniform distribution within the specified range, as shown in Equation (25).

$$\tilde{p}^i_{ejl} \in [L_{jl}, U_{jl}] \tag{25}$$

## *4.2. Decoding*

Section 4.1 initializes the solution population and scenario population through encoding, and decoding these populations can yield a complete scheduling solution. When decoding the scheduling decision gene $a^i_{ej}$, the ceiling part $\lceil a^i_{ej} \rceil$ represents the selected machine, and the decimal part $a^i_{ej} - \lfloor a^i_{ej} \rfloor$ in ascending order represents the sequence relationship on the machine. For the suspension shift decision variable $b^i_{ej}$, $b^i_{ej} = 0$ indicates that the operation can process normally in shift $j$, and $b^i_{ej} = 1$ indicates that the operation is suspended in shift $j$. The scenario gene $\tilde{p}^i_{ejl}$ represents the processing time of job $j$, which does not need to be decoded and can be directly used. Therefore, decoding $(a, b, p)$ yields the complete schedule solution.

## *4.3. Individual Evaluation*

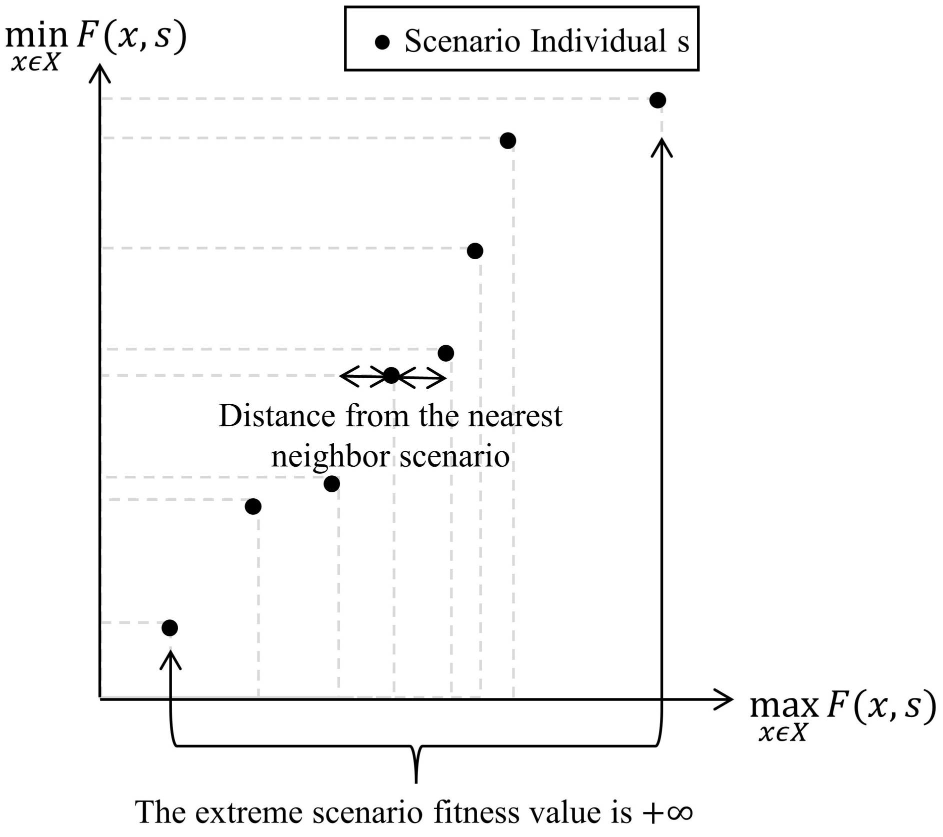
When facing uncertainty in processing times, due to the lack of specific probability distribution information, we can only rely on the upper and lower bounds of processing times to make decisions. Considering the differences in risk identification and management capabilities among decision-makers, their focus on performance evaluation metrics also varies. This paper introduces two evaluation metrics: the average criterion and the pessimistic criterion. These metrics aim to provide decision-makers with a more comprehensive information framework that better reflects the differences in risk preferences during the decision-making process. Through multidimensional evaluation, this approach enhances the adaptability and accuracy of decision-making in uncertain environments.

### 4.3.1. Average Criterion Individual Evaluation Method

The average criterion guides decision-making by considering the overall effect across all possible scenarios. It calculates the expected values of all potential outcomes to find the solution that performs best on average. This approach is especially suitable for risk-neutral decision-makers. Specifically, for the set of all possible processing time scenarios $S$ and the set of all feasible solutions $X$, the objective function $F(x, s)$ describes the weighted sum of earliness and tardiness for a particular solution $x$ under a specific processing time scenario $s$. The fitness value fitness$_x$ of solution $x$ can be determined by calculating the unweighted average of its objective function across all scenarios, as shown in Equation (26).

$$fitness_x = \frac{\sum_{s \in S} F(x, s)}{|S|} \tag{26}$$

Within the dual-space co-evolutionary algorithm framework, the evolution objective of the scenario population should focus on enhancing diversity. This diversity ensures that the algorithm maintains the robustness and effectiveness of its solutions when facing a wide range of changing environments. To make the scenario population more diverse, the fitness value of a scenario individual is transformed into its contribution to the population's diversity. This is determined by calculating its distance from other scenarios. The method of calculating the distance is based on feature generation, where scenario individual $s$ is mapped in a two-dimensional space based on two relevant features, as shown in Figure 5: one is the maximum value of the objective function revealed by the solution population $X$, $\max_{x \in X} F(x,s)$, and the other is the minimum value $\min_{x \in X} F(x,s)$. For each feature, scenarios representing extreme values are given high fitness values to encourage expanding the "space" occupied by the population. For the remaining scenarios, the two nearest scenarios are identified, and the product of their distances is calculated. The fitness value of a scenario is the maximum product of the distances along the two axes, giving preference to scenarios that fill gaps in the population space.



$\min_{x \epsilon X} F(x,s)$
● Scenario Individual s
Distance from the nearest neighbor scenario
$\max_{x \epsilon X} F(x,s)$
The extreme scenario fitness value is $+\infty$

**Figure 5.** Calculation of scenario population fitness values and distance quantification.

The pseudo-code for Algorithm 1, which calculates the scenario fitness value, is shown below.

In conclusion, the evaluation of individuals within a population using an average criterion is designed to identify the solution that exhibits the optimal average performance across various scenarios throughout the evolutionary process. The objective of evaluating scenarios is to assemble a set of scenarios that collectively exert the most diverse influence on the solutions, as illustrated in Figure 6. The star signs represent individuals in each population. Consequently, the evolution of solutions is framed as a minimization problem, where individuals $x \in X$ from the solution population are ranked in non-decreasing order based on their fitness values. Conversely, the evolution of scenarios is considered a maximization problem, necessitating that individuals $s \in S$ from the scenario population be ranked in non-increasing order of their fitness values.

**Algorithm 1** Pseudo-code for the calculation of scenario fitness values

```
Input: Solution population $X$, scenario population $S$, and the $|X| \times |S|$ matrix $F(x,s)$
Output: The fitness values of all individuals in the scenario population $S$
1: Initialization: Lists $B$ and $W$ of the $|S|$ layer, two-axis distance values $Bdist_s$ and $Wdist_s$
   of scenario $s$, and adaptability value $fitness_s$ of scenario $s$.
2: for j ← 1 to |S| do:
3:     $s \leftarrow S^j$;
4:     $B^j_{\text{best}} \leftarrow \min_{x \in X} F(x,s)$;
5:     $B^j_{\text{id}} \leftarrow s$;
6:     $W^j_{\text{worst}} \leftarrow \max_{s \in S} F(x,s)$;
7:     $W^j_{\text{id}} \leftarrow s$;
8: end for
9: sort $B$ in ascending order by best
10: sort $W$ in ascending order by worst
11: for j ← 2 to $|S| - 1$ do:
12:     $Bdist_{B^j_{\text{id}}} \leftarrow (B^{j+1}_{\text{best}} - B^j_{\text{best}}) \times (B^j_{\text{best}} - B^{j-1}_{\text{best}})$;
13:     $Wdist_{W^j_{\text{id}}} \leftarrow (W^j_{\text{worst}} - W^{j-1}_{\text{worst}}) \times (W^j_{\text{worst}} - W^{j-1}_{\text{worst}})$;
14: end for
15: for all $s \in S$ do:
16:     if $(s = B^1_{\text{id}})$ or $(s = B^{|S|}_{\text{id}})$ or $(s = W^1_{\text{id}})$ or $(s = W^{|S|}_{\text{id}})$ then:
17:         $fitness_s \leftarrow +\infty$;
18:     else:
19:         $fitness_s \leftarrow \max(Bdist_s, Wdist_s)$;
20:     end if
21: end for
```

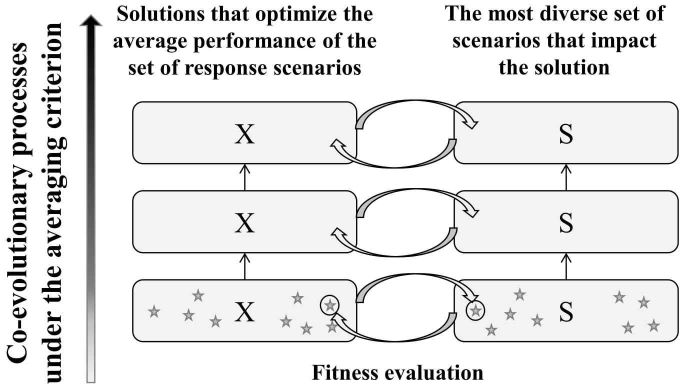

Co-evolutionary processes under the averaging criterion
Solutions that optimize the average performance of the set of response scenarios
The most diverse set of scenarios that impact the solution
X
S
X
S
X
S
Fitness evaluation

**Figure 6.** Flowchart of co-evolution under the average criterion.

### 4.3.2. Pessimistic Criterion Individual Evaluation Method

In decision theory, the pessimistic criterion, also known as the mini-max criterion, is a conservative decision-making method that is applied under conditions of uncertainty. This criterion assumes that the worst-case scenario will occur and selects the decision option that results in the least possible loss or minimizes the worst-case outcome. This approach is particularly suitable for decision-makers who are highly sensitive to risk because it minimizes potential maximum loss by considering all possible negative outcomes and choosing the option that guarantees the relatively best result in the worst-case scenario. Therefore, the fitness value $\text{fitness}_x$ of solution $x$ can be determined by calculating its worst-case objective function across all scenarios, as shown in Equation (27).

$$fitness_x = \max_{s \in S} F(x,s) \tag{27}$$

However, in the two-stage hybrid flow shop scheduling problem with suspension shifts in this paper, there are sequential constraints between processes. The completion time of each job cannot be expressed as a linear combination of the processing times $\{p_{11}, p_{21}, \ldots, p_{n1}, p_{12}, p_{22}, \ldots, p_{n2}\}$ of all jobs. Therefore, it is impossible to derive effective information about scenarios that would result in extreme performance values for the solution, making the search space for scenarios the entire feasible domain. According to Lemmas 1 and 2, Equation (28) can be used as the fitness function for evaluating scenario $s$.

$$fitness_s = \underset{x \in X}{min} F(x,s) \tag{28}$$

**Lemma 1** ([30])**.** *$\exists x^* \in X$, $s^* \in S : F(x^*,s) \leq F(x^*,s^*) \leq F(x,s^*)$ , thus, $\forall x \in X$, $s \in S$, $\min_{x \in X} \max_{s \in S} F(x,s) = \max_{s \in S} \min_{x \in X} F(x,s)$ and vice versa.*

**Lemma 2** ([30])**.** *If $(x_1, s_1)$ is a solution to the $\min_{x \in X} \max_{s \in S} F(x,s)$ problem and $(x_2, s_2)$ is a solution to the $\max_{s \in S} \min_{x \in X} F(x,s)$ problem, then $(x_1, s_2)$ is simultaneously a solution to both the $\min_{x \in X} \max_{s \in S} F(x,s)$ and $\max_{s \in S} \min_{x \in X} F(x,s)$ problems.*

In summary, with the pessimistic criterion, solution evaluation focuses on finding the best solution in the worst-case scenario. Meanwhile, scenario evaluation aims to identify the scenario that causes the worst solution performance, as shown in Figure 7. Therefore, the evolution of solutions is a minimization problem, and the individuals $x \in X$ in the solution population should be sorted in non-decreasing order of their fitness values. In contrast, the evolution of scenarios is a maximization problem, and the individuals $s \in S$ in the scenario population should be sorted in non-increasing order of their fitness values.

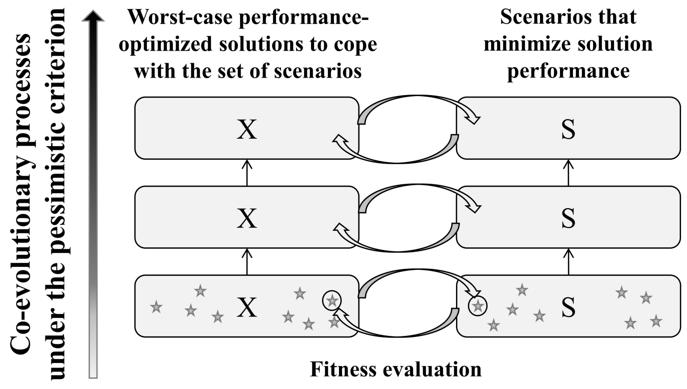

Co-evolutionary processes under the pessimistic criterion
Worst-case performance-optimized solutions to cope with the set of scenarios
Scenarios that minimize solution performance
X
S
X
S
X
S
Fitness evaluation

**Figure 7.** Flowchart of co-evolution under the pessimistic criterion.

### 4.4. Evolutionary Operations

In the DCE-BRKGA, an elite strategy, mutation, and crossover are used to generate new populations $X^{i+1}$ and $S^{i+1}$. The elite strategy is used to retain the top $P_e$ individuals, while mutation and crossover are used to generate $P - P_e$ new candidate individuals. These individuals form the new generation, which has a population of size $P$. This process is repeated until the termination condition is met.

#### 4.4.1. Elite Strategy

Since the individuals in the population were already sorted by fitness values in Section 4.3, the top $P_e$ individuals are the elite individuals $V_e^i$, and the remaining individuals are the non-elite individuals $Y_e^i$. The $P_e$ elite individuals are retained in the next-generation population.

#### 4.4.2. Mutation

We generate $P_m$ mutated individuals in the same way as the initialization in Section 4.1 and retain them in the next-generation population.

### 4.4.3. Crossover

After the mutation operation, crossover is performed to generate new individuals $W_e^i$ for the next-generation population. To maintain the population size in each generation, $P - P_e - P_m$ crossover individuals need to be generated. Crossover individuals $W_e^i = [w_{e1}^{i+1}, w_{e2}^{i+1}, \cdots, w_{en}^{i+1}]$ are generated, where each gene $w_{ej}^{i+1}$ is selected from the elite individual gene $v_{ej}^i$ and non-elite individual gene $y_{ej}^i$. This selection is controlled by the biased elite probability $\rho \in [0, 1]$, which is predetermined. The gene $w_{ej}^{i+1}$ can be obtained by Equation (29).

$$w_{ej}^{i+1} = \begin{cases} v_{ej}^i & if\ r(j) \le \rho \\ y_{ej}^i & otherwise \end{cases} \tag{29}$$

In Equation (29), $r(j)$ is a random number between (0,1). If the generated random number is less than $\rho$, the elite individual gene is chosen; otherwise, the non-elite individual gene is chosen. When performing crossover on the solution population and scenario population, the scheduling decision part of each individual's gene is crossed using the method, while the suspension shift decision part of the gene directly chooses the elite individual's gene.

## 5. Computational Experiments, Results, and Discussion

Under conditions of processing uncertainty, simulation tests are conducted on the two-stage hybrid flow shop scheduling problem with suspension shifts. The parameters of the generated data instances are as follows: The number of machines per stage $m \in \{2, 3, 4, 5, 6, 7\}$, with $m_1$ and $m_2$ both equal to $m$. The number of jobs $n \in \{10, 15, 20, 25, 30, 35\}$. The lower bound of the processing time for job $j$ in stage $l$ follows a uniform distribution $L_{jl} \in [L_{jl}, \gamma_1 L_{jl}]$, where $L$ represents the minimum processing time per stage for a job; to ensure at least four days of workload, $L$ is set to $96m/n$. The upper bound of the processing time for job $j$ in stage $l$ follows a uniform distribution $U_{jl} \in [L_{jl}, (1 + \gamma_2)L_{jl}]$, with parameters $\gamma_1 = 2$ and $\gamma_2 = 1.5$ controlling the relative ranges of the lower and upper bounds. To test the impact of due dates, the due date $d_j$ of job $j$ follows a uniform distribution $(\mu - \mu R/2, \mu + \mu R/2)$, where $\mu = (1 - T)E[C_{\max}]$ and $E[C_{\max}] = \frac{1}{2m} \sum_{j=1}^{n} \sum_{j=1}^{2} \frac{L_{jl}+U_{jl}}{2}$. The parameter $T = 0.3$ is the tardiness factor, and $R \in \{0.6, 1.2\}$ is the relative range of the due date window. When $R = 0.6$, the due dates are tight, and when $R = 1.2$, the due dates are relaxed. The weights for job earliness and tardiness follow a discrete uniform distribution in the range $[a, b]$. The specific parameter values are shown in Table 2. There are 72 possible combinations. From these parameter combinations, 28 different scale combinations are selected to generate 28 data instances for experimental testing. The specific data instance numbers and corresponding parameter combinations can be seen in Table 3.

**Table 2.** Parameters of test data instances.

| Parameter | Experimental Value | Parameter Type |
|---|---|---|
| Number of Machines per Stage | 2, 3, 4, 5, 6, 7 | 6 |
| Number of Jobs | 10, 15, 20, 25, 30, 35 | 6 |
| Lower Bound of Processing Time | Uniform distribution of $L_{jl} \in [\iota_{jl}, \gamma_1 L_{jl}]$, where $L = \frac{96m}{n}, \gamma_1 = 2$ | 1 |
| Upper Bound of Processing Time | Uniform distribution of $U_{jt} \in [L_{jt}, (1 + \gamma_2)L_{jt}]$, where $\gamma_2 = 1.5$ | 1 |
| Due date | $d_j \in (\mu - \frac{\mu R}{2}, \mu + \frac{\mu R}{2})$, $\mu = (1 - T)E[C_{\max}]$, $E[C_{\max}] = \frac{1}{2m} \sum_{j=1}^{n} \sum_{l=1}^{n} \frac{L_{jl}+U_{jl}}{2}$, $T = 0.3, R \in \{0.6, 1.2\}$ | 2 |
| Earliness\Tardiness Weight | $\alpha_j, \beta_j$ follow a discrete uniform distribution over the range [1, 9] | 1 |

**Table 3.** Results of the analysis of the evolution of the solution space.

| | | Average Criterion | | | | | Pessimistic Criterion | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Data Instance Number** | **Scale ($n \times m_1 \times m_2 \times R$)** | **Degree of Improvement of Solution** | **Optimal Fitness Value** | **Coefficient of Variation** | **Average Running Time (s)** | **Optimal Shifts for Suspension** | **Degree of Improvement of Solution** | **Optimal Fitness Value** | **Coefficient of Variation** | **Average Running Time (s)** | **Optimal Shifts for Suspension** |
| 1 | $10 \times 2 \times 2 \times 0.6$ | 61.5% | 588.1 | 7.3% | 16.2 | 0 | 59.2% | 611 | 11.8% | 13.5 | 0 |
| 2 | $10 \times 2 \times 2 \times 1.2$ | 78.4% | 396.2 | 22.7% | 25.3 | 2 | 70.2% | 542.1 | 14.9% | 24.7 | 0 |
| 3 | $10 \times 3 \times 3 \times 0.6$ | 59.7% | 824.4 | 16.6% | 24.2 | 0 | 65.9% | 1042.8 | 11.2% | 20.2 | 0 |
| 4 | $10 \times 3 \times 3 \times 1.2$ | 75.3% | 515.2 | 27.4% | 39.8 | 2 | 74.1% | 581.8 | 13.7% | 43.2 | 2 |
| 5 | $10 \times 4 \times 4 \times 0.6$ | 55.9% | 679.3 | 23.9% | 32.3 | 1 | 81.2% | 787.9 | 28.4% | 36.5 | 1 |
| 6 | $10 \times 4 \times 4 \times 1.2$ | 87.7% | 306.3 | 26.5% | 41.9 | 2 | 64.7% | 570 | 9.4% | 43.8 | 3 |
| 7 | $15 \times 2 \times 2 \times 0.6$ | 57.7% | 1087.4 | 9.1% | 51.5 | 1 | 65.7% | 1196.1 | 5.7% | 54.1 | 1 |
| 8 | $15 \times 2 \times 2 \times 1.2$ | 80.7% | 467.1 | 19.1% | 102.2 | 1 | 78.3% | 552 | 13.9% | 72 | 0 |
| 9 | $15 \times 3 \times 3 \times 0.6$ | 59.8% | 1125.6 | 13.6% | 92.9 | 0 | 61.2% | 1257.1 | 8.9% | 106.9 | 1 |
| 10 | $15 \times 3 \times 3 \times 1.2$ | 79.2% | 619.7 | 12.7% | 134.1 | 4 | 82.2% | 621.9 | 13.1% | 102.6 | 2 |
| 11 | $15 \times 4 \times 4 \times 0.6$ | 75.5% | 1026.2 | 19.2% | 124.6 | 1 | 77.9% | 1087.7 | 21.2% | 105.6 | 0 |
| 12 | $15 \times 4 \times 4 \times 1.2$ | 81.8% | 657.4 | 12.2% | 164.3 | 0 | 77.2% | 821.4 | 11.4% | 108.5 | 0 |
| 13 | $20 \times 2 \times 2 \times 0.6$ | 61.3% | 1409 | 6.8% | 139.4 | 1 | 56.2% | 1518.6 | 4.5% | 143.9 | 0 |
| 14 | $20 \times 2 \times 2 \times 1.2$ | 87.3% | 630 | 14.4% | 251.3 | 1 | 80.5% | 797.7 | 14.1% | 214.8 | 3 |
| 15 | $20 \times 3 \times 3 \times 0.6$ | 69.1% | 1417.1 | 7.4% | 244.5 | 1 | 56.6% | 1620.6 | 6.5% | 216.9 | 1 |
| 16 | $20 \times 3 \times 3 \times 1.2$ | 69.9% | 1123.1 | 11.7% | 266.3 | 2 | 74.8% | 1364.4 | 11.6% | 286.2 | 0 |
| 17 | $20 \times 4 \times 4 \times 0.6$ | 58.3% | 1763 | 11.7% | 264.3 | 0 | 71.3% | 1847.6 | 8.7% | 328.7 | 1 |
| 18 | $20 \times 4 \times 4 \times 1.2$ | 77.2% | 896.9 | 11.6% | 483.7 | 2 | 78.0% | 967.9 | 16.9% | 362.5 | 2 |
| 19 | $25 \times 5 \times 5 \times 0.6$ | 66.1% | 2211.6 | 7.0% | 693.1 | 1 | 68.6% | 2335.4 | 4.5% | 773.5 | 0 |
| 20 | $25 \times 5 \times 5 \times 1.2$ | 78.2% | 1270.5 | 11.2% | 936.5 | 0 | 78.1% | 1498.5 | 13.5% | 937.6 | 2 |
| 21 | $25 \times 6 \times 6 \times 0.6$ | 72.8% | 1544.6 | 10.7% | 821.3 | 0 | 71.9% | 1757.8 | 6.1% | 930.8 | 0 |
| 22 | $25 \times 6 \times 6 \times 1.2$ | 75.3% | 1437.6 | 11.0% | 1078 | 0 | 79.1% | 1543.4 | 11.6% | 1145.7 | 2 |
| 23 | $30 \times 6 \times 6 \times 0.6$ | 74.9% | 2773.5 | 9.4% | 1580.7 | 1 | 71.9% | 3071.8 | 7.1% | 1488.4 | 0 |
| 24 | $30 \times 6 \times 6 \times 1.2$ | 76.6% | 1183.3 | 7.0% | 2374.3 | 2 | 82.9% | 1241.6 | 10.9% | 2113.5 | 1 |
| 25 | $30 \times 7 \times 7 \times 0.6$ | 67.7% | 2660.2 | 6.8% | 2142.7 | 0 | 73.0% | 2669.1 | 9.6% | 1826.2 | 0 |
| 26 | $30 \times 7 \times 7 \times 1.2$ | 78.8% | 1591 | 8.4% | 2760.7 | 1 | 78.6% | 1904.6 | 6.9% | 2582.4 | 1 |
| 27 | $35 \times 7 \times 7 \times 0.6$ | 68.8% | 2696 | 5.1% | 3074.1 | 0 | 64.7% | 2974.7 | 5.5% | 3475.5 | 0 |
| 28 | $35 \times 7 \times 7 \times 1.2$ | 80.2% | 1515.8 | 9.2% | 6082.5 | 1 | 81.9% | 1817.1 | 7.2% | 6274.4 | 2 |
| Average | | 72.0% | 1229.1 | 12.8% | 858.7 | 1.0 | 72.4% | 1378.7 | 11.0% | 851.2 | 0.9 |

The performance of the BRKGA algorithm is highly sensitive to its parameters: population size $P$, elite individual proportion $P_e$, mutant individual proportion $P_m$, and elite gene inheritance probability $\rho$. Referencing the parameter choices recommended by Gonçalves [31], we set $P_e$ for the DCE-BRKGA dual-space parallel evolution to 0.2, $P_m$ is set to 0.1, and $\rho$ is set to 0.7. The population size is set to $P = 2\sqrt{n \times (m_1 + m_2)}$, where $n$ is the total number of jobs, and $m_1$ and $m_2$ are the numbers of machines in the first and second stages, respectively. The stopping criterion is based on $2n$ non-improving iterations.

All experiments are conducted on a desktop computer with an AMD Ryzen 9 5900X 12-Core (3.70 GHz; AMD, Santa Clara, CA, USA) CPU and 32 GB RAM, using Python 3.9 and Gurobi 11.0.0 as the experimental tools.

### *5.1. Solution Evolution*

To evaluate the algorithm's ability to generate high-quality solutions, we can analyze the performance improvement during iterations and the performance differences across multiple runs. Figure 8 shows the iterative process for data Instance 11, illustrating the changes in the fitness values of the best solution in each generation, making it easy to observe the evolution and convergence. Despite fluctuations caused by changes in the scenario population during evolution, the fitness values under both evaluation criteria significantly improved and eventually stabilized. To quantify the performance improvement, we analyze the iteration process that generated the optimal solution. Using the scenario population from the final generation of that iteration, we calculate the initial solution's fitness value $f_{initial}$ and the final optimal solution's fitness value $f_{final}$. These values are used to compute the improvement during the iteration, as shown in Equation (30). Table 3 records the improvement for each data instance under different evaluation metrics, showing consistent improvement across both metrics, with an average improvement of about 72%.

$$Degree\ of\ improvement\ of\ the\ solution = \frac{f_{final} - f_{initial}}{f_{initial}} \times 100\% \tag{30}$$

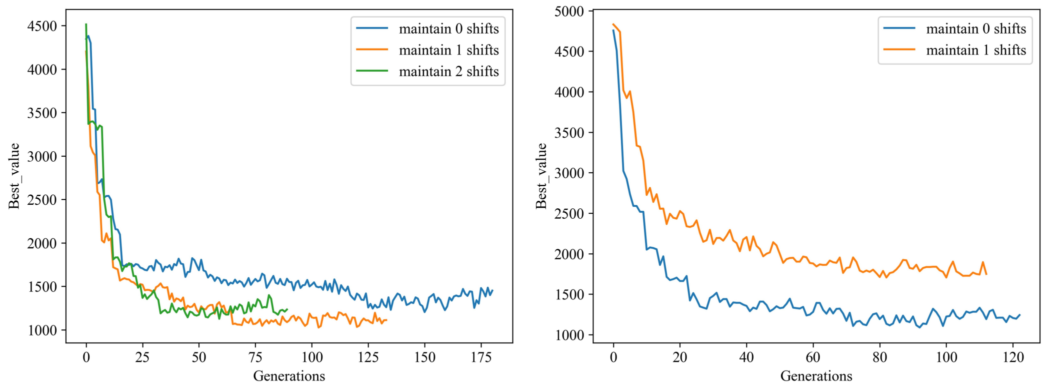


**Figure 8.** Evolution of the solution's fitness values (Instance 11, with the average criterion on the left and the pessimistic criterion on the right).

In this study, each data instance is run ten times to ensure result reliability, and we record the fitness values of the final solutions for each run. The lowest fitness value among these runs is selected as the optimal fitness value for each data instance. As shown in the "Optimal Fitness Value" column in Table 3 and Figure 9, the optimal fitness values obtained using the pessimistic criterion are generally higher than those obtained using the average criterion. This indicates that the pessimistic criterion tends to consider the worst-case scenarios, leading to relatively poorer fitness values.

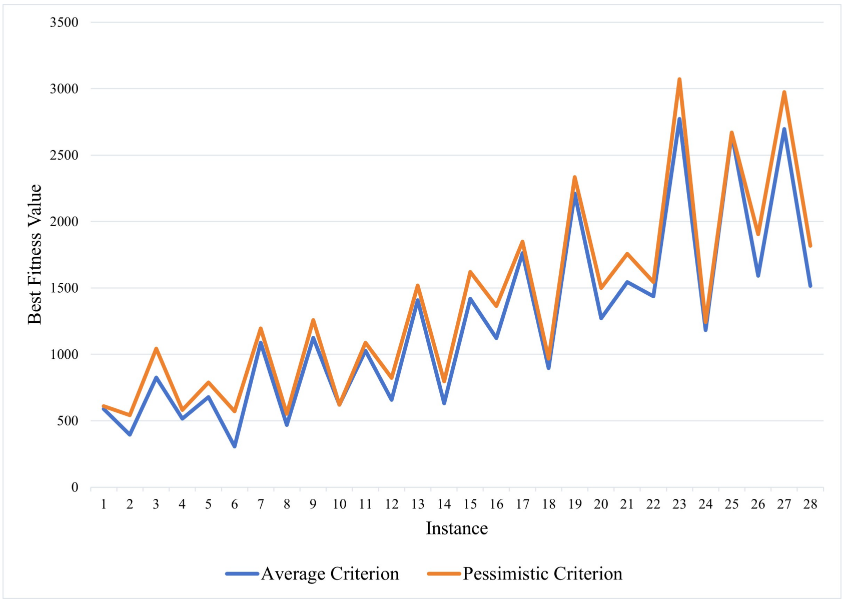


**Figure 9.** Optimal fitness values for the average and the pessimistic criterion.

To assess the algorithm's stability and reliability, we analyze the coefficient of variation of the fitness values from ten runs. The "Coefficient of Variation" column in Table 3 and Figure 10 show average values of 12.8% and 11.0% for the two evaluation criteria, indicating some variability between runs due to the different scenario populations generated each time. Specifically, when the number of jobs $n$ is 10, the coefficient of variation is unstable, sometimes exceeding 20%. This is likely because with fewer jobs, the processing time of each job becomes more critical, and minor adjustments to solutions can lead to significant changes in fitness values.

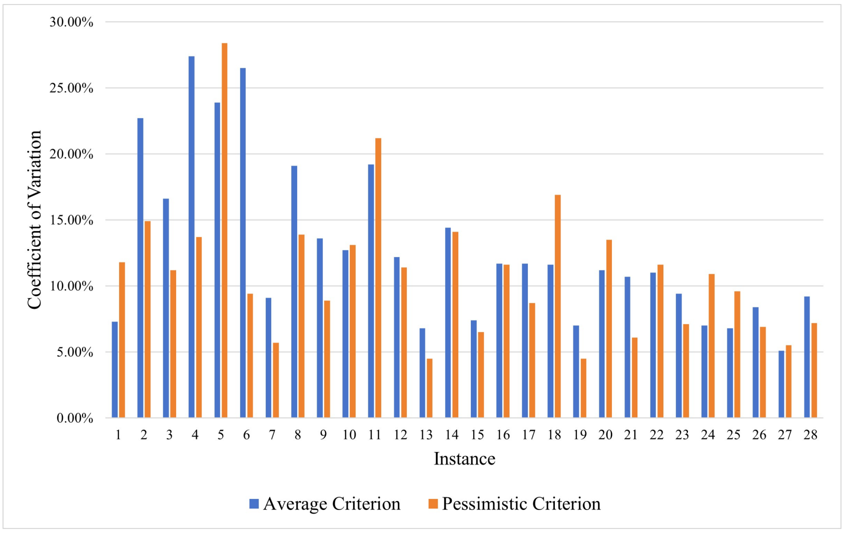


**Figure 10.** Coefficients of variation for the average and pessimistic criterion.

Table 3 provides a quantitative analysis of the average run times, revealing that run times are significantly affected by the scale of the data instances. As the number of jobs and machines increases, so does the run time. Additionally, when the due dates are more relaxed, the run time tends to increase. This may be because relaxed due dates imply ample capacity, requiring more suspension shifts to meet production needs. To further verify this observation, Table 3 also records the number of suspension shifts corresponding to the optimal solution. The data show that in instances with $R = 1.2$, the optimal solutions generally require more suspension shifts than those with $R = 0.6$, supporting the above findings.

### *5.2. Algorithm Comparison Experiments*

To further verify the performance of the algorithm in terms of the quality of the generated solution, this paper compares the BRKGA with the Gurobi solver and the RKGA under deterministic scenarios.

#### 5.2.1. Comparative Experiments with the BRKGA and Gurobi

Since this problem is NP-hard, Gurobi can only solve relatively small data instances. This paper designs 12 small data instances so that Gurobi can achieve optimal solutions within a reasonable time. The method for generating data instances is similar to that in Section 5, with two machines per stage and the number of jobs $n \in \{4, 6, 8, 10, 12\}$; the other parameters are generated in the same way. Each data instance generates a fixed scenario within the processing time range. In this deterministic scenario, a comparative experiment is conducted between the BRKGA and Gurobi; we limit the suspension shifts to no more than one and compare the minimum objective values obtained by both methods.

As shown in Table 4, the final objective value obtained by the BRKGA is, on average, 7.08% higher than the optimal value obtained by Gurobi. In terms of the average run time, when $n = 12$ and $m_1 = m_2 = 2$, Gurobi takes over an hour to solve the problem, while the BRKGA completes it within 8 s. This demonstrates that the BRKGA can effectively solve the problem in a much shorter time.

**Table 4.** The BRKGA and Gurobi solve for the optimal value and average run time of the data instance.

| Scale ($n \times m_1 \times m_2 \times R$) | Optimal Solution Gurobi | BRKGA | Deviation | Average Run Time Gurobi | BRKGA |
|---|---|---|---|---|---|
| $4 \times 2 \times 2 \times 0.6$ | 960 | 979 | 1.98% | 0.02 | 0.18 |
| $4 \times 2 \times 2 \times 1.2$ | 719 | 784 | 9.04% | 0.03 | 0.18 |
| $6 \times 2 \times 2 \times 0.6$ | 441 | 467 | 5.90% | 0.36 | 0.71 |
| $6 \times 2 \times 2 \times 1.2$ | 338 | 362 | 7.10% | 0.34 | 0.70 |
| $8 \times 2 \times 2 \times 0.6$ | 894 | 965 | 7.94% | 1.29 | 2.05 |
| $8 \times 2 \times 2 \times 1.2$ | 517 | 564 | 9.09% | 2.06 | 1.73 |
| $10 \times 2 \times 2 \times 0.6$ | 917 | 998 | 8.83% | 242.80 | 3.55 |
| $10 \times 2 \times 2 \times 1.2$ | 837 | 898 | 7.29% | 283.02 | 2.96 |
| $12 \times 2 \times 2 \times 0.6$ | 498 | 521 | 4.62% | 3864.32 | 7.60 |
| $12 \times 2 \times 2 \times 1.2$ | 398 | 434 | 9.05% | 4510.70 | 7.87 |
| Average | 651.9 | 697.2 | 7.08% | 890.49 | 2.75 |

#### 5.2.2. Comparative Experiments with the BRKGA and RKGA

This paper compares the optimal solutions generated by the RKGA [32] and the BRKGA under two different fitness criteria in a fixed scenario. Comparative experiments are conducted on 14 data instances, with the results shown in Table 5 and Figure 11. The average deviation of the final best value of the BRKGA from that of the RKGA is −49.7% under the average criterion and −50.4% under the pessimistic standard. Furthermore, Figure 11 indicates that, compared to the average criterion, the pessimistic criterion results in a smaller variance in the deviation of the BRKGA relative to the RKGA across the 14 data instances. Although the main advantage of the proposed heuristic method is that it generates scenarios simultaneously, which is not evaluated in this experiment, the results support that the BRKGA under a co-evolutionary mechanism yields better solutions than the RKGA. The BRKGA's advantages become more apparent as the scale of the data instances increases.

In addition to comparing the optimal values of the two algorithms, their average run times were also compared, as shown in Table 6. It can be seen that under both evaluation criteria, the BRKGA's average run time is longer. This is because the RKGA is more likely to get trapped in local optima, leading to early termination of iterations. This reflects the BRKGA's advantage in balancing search depth and breadth. The BRKGA's elite strategy and biased selection strategy effectively balance the search breadth (exploring new potential solutions) and depth (optimizing known solutions), ensuring that the algorithm neither gets trapped in local optima too early nor fails to effectively approach the global optimum.

**Table 5.** Optimization of BRKGA and RKGA for solving data instances under different evaluation criteria.

| **Instance** | **Scale** | **Average Criterion** | | | **Pessimistic Criterion** | | |
|---|---|---|---|---|---|---|---|
| **Number** | **($n \times m_1 \times m_2 \times R$)** | **RKGA** | **BRKGA** | **Deviation** | **RKGA** | **BRKGA** | **Deviation** |
| 1 | $10 \times 2 \times 2 \times 0.6$ | 764.5 | 597.6 | −21.8% | 840.3 | 623.4 | −25.8% |
| 2 | $10 \times 2 \times 2 \times 1.2$ | 905.6 | 509.3 | −43.8% | 1106.9 | 589.5 | −46.7% |
| 3 | $10 \times 3 \times 3 \times 0.6$ | 1265.2 | 869.0 | −31.3% | 1647.8 | 854.2 | −48.2% |
| 4 | $10 \times 3 \times 3 \times 1.2$ | 1041.6 | 592.4 | −43.1% | 1338.0 | 740.6 | −44.6% |
| 5 | $10 \times 4 \times 4 \times 0.6$ | 1099.0 | 642.0 | −41.6% | 1785.4 | 811.7 | −54.5% |
| 6 | $10 \times 4 \times 4 \times 1.2$ | 1022.9 | 445.6 | −56.4% | 1381.2 | 447.9 | −67.6% |
| 7 | $15 \times 2 \times 2 \times 0.6$ | 2032.6 | 1145.5 | −43.6% | 1887.2 | 1259.3 | −33.3% |
| 8 | $15 \times 2 \times 2 \times 1.2$ | 1337.8 | 564.7 | −57.8% | 1629.2 | 659.6 | −59.5% |
| 9 | $15 \times 3 \times 3 \times 0.6$ | 2633.0 | 1143.3 | −56.6% | 2557.0 | 1315.8 | −48.5% |
| 10 | $15 \times 3 \times 3 \times 1.2$ | 1947.5 | 594.7 | −69.5% | 1751.7 | 750.8 | −57.1% |
| 11 | $15 \times 4 \times 4 \times 0.6$ | 2544.3 | 1194.8 | −53.0% | 2777.7 | 1258.7 | −54.7% |
| 12 | $15 \times 4 \times 4 \times 1.2$ | 2068.5 | 692.9 | −66.5% | 1513.9 | 775.3 | −48.8% |
| 13 | $20 \times 2 \times 2 \times 0.6$ | 2551.5 | 1483.0 | −41.9% | 2988.4 | 1582.8 | −47.0% |
| 14 | $20 \times 2 \times 2 \times 1.2$ | 2212.1 | 701.3 | −68.3% | 2502.0 | 752.0 | −69.9% |
| | Average | 1673.3 | 798.3 | −49.7% | 1836.2 | 887.3 | −50.4% |

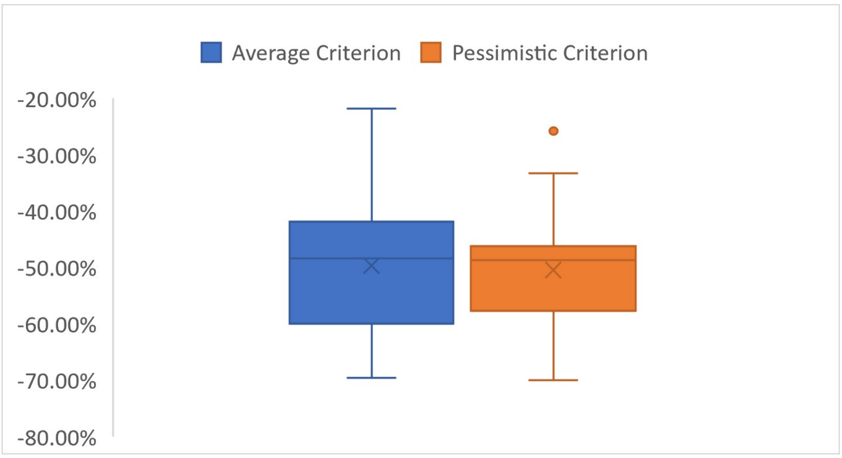


**Figure 11.** Relative deviation of the final optimal values sought by the BRKGA and the RKGA.

**Table 6.** Average run times of BRKGA and RKGA solving data instances under different evaluation criteria.

| **Instance** | **Scale** | **Average Criterion** | | **Pessimistic Criterion** | |
|---|---|---|---|---|---|
| **Number** | **($n \times m_1 \times m_2 \times R$)** | **RKGA** | **BRKGA** | **RKGA** | **BRKGA** |
| 1 | $10 \times 2 \times 2 \times 0.6$ | 7.6 | 11.6 | 6.6 | 11.9 |
| 2 | $10 \times 2 \times 2 \times 1.2$ | 10 | 16.2 | 9.1 | 21.8 |
| 3 | $10 \times 3 \times 3 \times 0.6$ | 10.8 | 20.8 | 12.8 | 20.3 |
| 4 | $10 \times 3 \times 3 \times 1.2$ | 13.4 | 28.9 | 15.3 | 26.3 |
| 5 | $10 \times 4 \times 4 \times 0.6$ | 13.4 | 22.7 | 11.8 | 35.8 |
| 6 | $10 \times 4 \times 4 \times 1.2$ | 16.3 | 37.1 | 18.9 | 38.3 |
| 7 | $15 \times 2 \times 2 \times 0.6$ | 20.1 | 64.4 | 24.8 | 47.7 |
| 8 | $15 \times 2 \times 2 \times 1.2$ | 32.1 | 71.3 | 30.6 | 121.1 |
| 9 | $15 \times 3 \times 3 \times 0.6$ | 31.4 | 82.5 | 30.7 | 88.9 |
| 10 | $15 \times 3 \times 3 \times 1.2$ | 43.3 | 101.1 | 41.1 | 101.9 |
| 11 | $15 \times 4 \times 4 \times 0.6$ | 54.2 | 151.3 | 40.3 | 112.9 |
| 12 | $15 \times 4 \times 4 \times 1.2$ | 47.2 | 210.4 | 49.9 | 115 |
| 13 | $20 \times 2 \times 2 \times 0.6$ | 63 | 173.9 | 47.5 | 150.2 |
| 14 | $20 \times 2 \times 2 \times 1.2$ | 75.7 | 196.9 | 63.7 | 211.3 |
| | Average | 31.3 | 84.9 | 28.8 | 78.8 |

### *5.3. Analysis Based on the EVPI and VSS Indicators*

The DCE-BRKGA is a heuristic algorithm, so its results, including solutions and scenarios, are not guaranteed to be optimal. When evaluated using the average criterion, the problem is classified as a stochastic optimization problem. For such problems, the EVPI and VSS are used to assess the benefits of the stochastic optimization solutions obtained by the DCE-BRKGA under uncertainty. This evaluation helps quantify the potential value and effectiveness of the algorithm's solutions when accounting for uncertainty.

Figure 12 illustrates the framework for calculating the EVPI and VSS. The EVPI is the difference between the optimal fitness value randomized adaptation (RA) obtained by the stochastic method and the "wait-and-see" (WS) value. The WS value represents the average performance metric of the optimal solutions for each scenario, assuming perfect information is available in advance. The DCE-BRKGA evolves based on two parallel BRKGAs, where solutions and scenarios co-evolve. To obtain comparable metrics, a similar heuristic algorithm, a single BRKGA, is used to generate the "wait-and-see" solutions. Unlike the two parallel BRKGAs, the single BRKGA evolves only the solution population, assuming fixed (deterministic) scenarios. The VSS compares the optimal fitness value RA obtained by the stochastic method with the expected value from using the expected solution (EEV) considering only the average scenario. A single-space BRKGA generates the expected result, where the processing time for the average scenario is derived from the mean of the processing times for all scenarios.

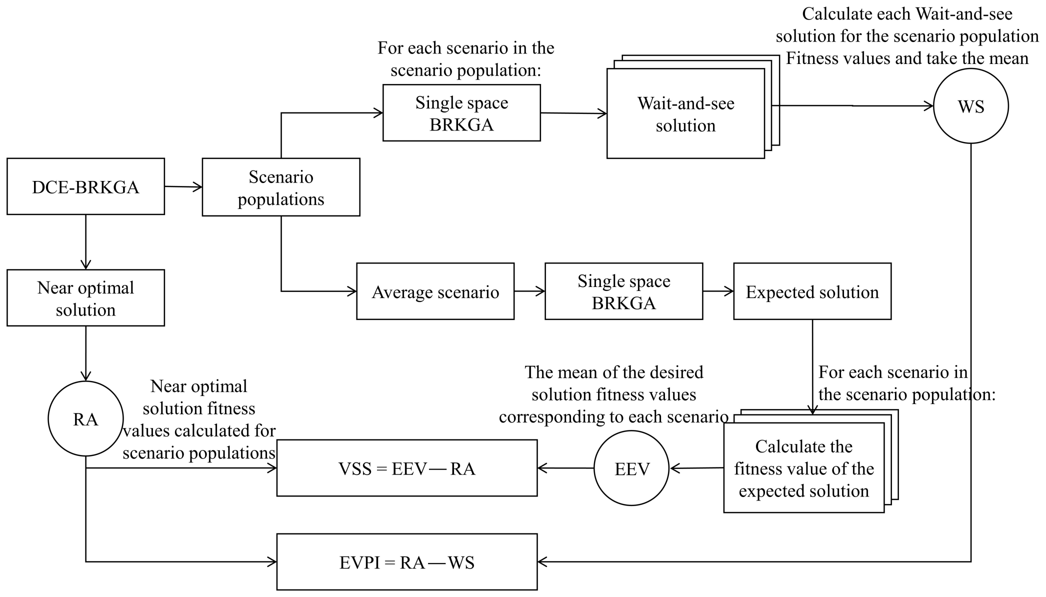

Calculate each Wait-and-see solution for the scenario population Fitness values and take the mean
For each scenario in the scenario population:
Single space BRKGA
Wait-and-see solution
WS
DCE-BRKGA
Scenario populations
Near optimal solution
Average scenario
Single space BRKGA
Expected solution
For each scenario in the scenario population:
The mean of the desired solution fitness values corresponding to each scenario
RA
Near optimal solution fitness values calculated for scenario populations
VSS = EEV—RA
EEV
Calculate the fitness value of the expected solution
EVPI = RA—WS

**Figure 12.** Framework for calculating EVPI and VSS.

To further verify the potential value and efficiency of the DCE-BRKGA, the expected value of perfect information as a percentage ($EVPI_{\%}$) is used to evaluate the value of obtaining perfect information. This value is defined as the degree of deviation between the RA obtained by the DCE-BRKGA and the WS obtained by the "wait-and-see" method and is calculated as:

$$EVPI_{\%} = \frac{\mathrm{RA} - \mathrm{WS}}{\mathrm{WS}} \times 100\%, \tag{31}$$

The value of the stochastic solution as a percentage ($VSS_{\%}$) represents the degree of deviation of the expected value with respect to the RA and is calculated as:

$$VSS_{\%} = \frac{\mathrm{EEV} - \mathrm{RA}}{\mathrm{RA}} \times 100\%, \tag{32}$$

This study uses 28 data instances to calculate the EVPI and VSS, and the results are shown in Table 7 and Figure 13. For the EVPI, the results indicate that if uncertainty is eliminated, potential improvements of 2.4% to 20.3% can be expected, assuming the scenario population is representative and decisions are made. This value is very useful for manufacturers as it helps them understand the value of investing in better prediction methods.

Regarding the VSS, this metric directly measures the impact of using a stochastic method compared to a similar deterministic method. The results show that the consideration of uncertainty in the decision-making process can achieve additional value gains of 0.9% to 69.9%. This finding highlights the importance and potential benefits of incorporating uncertainty into decision-making.

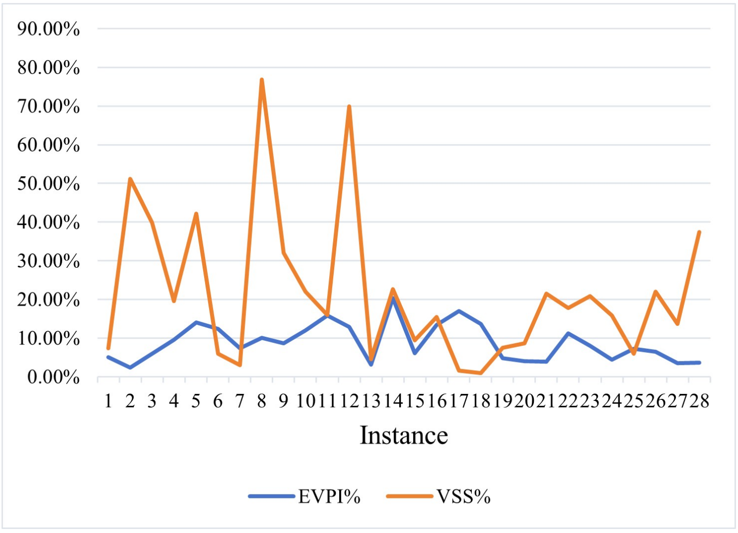


**Figure 13.** Results of the EVPI and VSS calculations.

**Table 7.** Calculation of EVPI and VSS.

| Instance Number | RA | WS | EEV | EVPI | $EVPI_{\%}$ | VSS | $VSS_{\%}$ |
|---|---|---|---|---|---|---|---|
| 1 | 588.1 | 559.3 | 631.6 | 28.8 | 5.1% | 43.5 | 7.4% |
| 2 | 396.2 | 386.9 | 598.9 | 9.3 | 2.4% | 202.7 | 51.2% |
| 3 | 824.4 | 778.2 | 1152.3 | 46.2 | 5.9% | 327.9 | 39.8% |
| 4 | 515.2 | 470.2 | 616.1 | 45.0 | 9.6% | 100.9 | 19.6% |
| 5 | 679.3 | 595.9 | 965.9 | 83.4 | 14.0% | 286.6 | 42.2% |
| 6 | 306.3 | 272.6 | 324.4 | 33.7 | 12.4% | 18.1 | 5.9% |
| 7 | 1087.4 | 1012.2 | 1120.4 | 75.2 | 7.4% | 33.0 | 3.0% |
| 8 | 467.1 | 424.2 | 825.8 | 42.9 | 10.1% | 358.7 | 76.8% |
| 9 | 1125.6 | 1036.1 | 1486.0 | 89.5 | 8.6% | 360.4 | 32.0% |
| 10 | 619.7 | 553.1 | 755.9 | 66.6 | 12.0% | 136.2 | 22.0% |
| 11 | 1026.2 | 885.8 | 1190.9 | 140.4 | 15.9% | 164.7 | 16.0% |
| 12 | 657.4 | 582.5 | 1117.1 | 74.9 | 12.9% | 459.7 | 69.9% |
| 13 | 1409 | 1366.6 | 1472.7 | 42.4 | 3.1% | 63.7 | 4.5% |
| 14 | 630 | 523.7 | 772.2 | 106.3 | 20.3% | 142.2 | 22.6% |
| 15 | 1417.1 | 1336.2 | 1550.1 | 80.9 | 6.1% | 133.0 | 9.4% |
| 16 | 1123.1 | 990.5 | 1296.0 | 132.6 | 13.4% | 172.9 | 15.4% |
| 17 | 1763 | 1507.1 | 1791.7 | 255.9 | 17.0% | 28.7 | 1.6% |
| 18 | 896.9 | 789.1 | 904.8 | 107.8 | 13.7% | 7.9 | 0.9% |
| 19 | 2211.6 | 2111.1 | 2378.5 | 100.5 | 4.8% | 166.9 | 7.5% |
| 20 | 1270.5 | 1222.2 | 1380.7 | 48.3 | 4.0% | 110.2 | 8.7% |
| 21 | 1544.6 | 1487.0 | 1877.4 | 57.6 | 3.9% | 332.8 | 21.5% |
| 22 | 1437.6 | 1293.3 | 1694.0 | 144.3 | 11.2% | 256.4 | 17.8% |
| 23 | 2773.5 | 2568.6 | 3353.7 | 204.9 | 8.0% | 580.2 | 20.9% |
| 24 | 1183.3 | 1133.2 | 1370.6 | 50.1 | 4.4% | 187.3 | 15.8% |
| 25 | 2660.2 | 2478.5 | 2816.5 | 181.7 | 7.3% | 156.3 | 5.9% |
| 26 | 1591 | 1494.1 | 1940.6 | 96.9 | 6.5% | 349.6 | 22.0% |
| 27 | 2696 | 2605.9 | 3066.2 | 90.1 | 3.5% | 370.2 | 13.7% |
| 28 | 1515.8 | 1462.3 | 2082.7 | 53.5 | 3.7% | 566.9 | 37.4% |

### *5.4. Scenario Evolution*

This section aims to evaluate the quality of the final scenario population obtained using the average criterion, focusing on diversity and representativeness. In Section 5.1, we discussed that the fluctuations in solution fitness values (see Figure 8) when using the average criterion are caused by the non-monotonic evolution of the scenario population. Due to strategies to increase the scenario population diversity, scenarios are ranked based on their differential impact on the solutions. This means that the addition or removal of a single individual can significantly change the fitness of another individual. However, in this study, the fitness of each scenario is quantified to guide the population towards certain features, with the focus being on the overall structure of the population rather than the fitness of individual scenarios. Therefore, when considering the quality of the scenario population, two key features are: diversity—the goal is to obtain scenarios that have different impacts on the solutions; representativeness—since the scenario population cannot cover all possible scenarios, it is essential to ensure that the selected scenarios reflect the performance of the solutions across all potential scenarios.

#### 5.4.1. Diversity

The diversity of the scenario population relates to the calculation of the scenario fitness values introduced in Section 4.3.1. Figure 14 uses the previously introduced two-dimensional coordinate system, as shown in Figure 5, where each scenario in the population is mapped based on the worst and best fitness values it “causes” in the solution population. This figure compares the initial and final generations of the scenario population as mapped within the final solution population. Compared to the initial scenario population (blue triangles), the final generation (red circles) occupies a larger “space”, with greater distances between scenarios, indicating higher diversity. The population’s skewed distribution is because the two features used to map scenarios are correlated; higher worst fitness values tend to accompany higher best fitness values. Considering that the solution population tends to converge towards similarly well-performing solutions, we do not expect to see a single scenario simultaneously adversely affecting one solution while benefiting another in the final generation.

Based on this mapping, population diversity can be measured from two perspectives: the range occupied in the space and the dispersion among scenarios. To quantitatively assess the space and dispersion, this study records the extremum and standard deviations for the two features. The metrics for the initial and final generations of the 28 data instances are detailed in Table 8.

To compare the initial and final generations of the scenario population in terms of occupied space and dispersion, this study introduces additional evaluation metrics: the extremum deviation ratio (EDR) and the standard deviation ratio (SDR). The EDR measures the expansion of the final generation’s range on the worst- and best-fitness-value axes compared to the initial generation. The SDR measures the increase in dispersion of the final generation relative to the initial generation.

Table 9 and Figure 15 show that, on average, the final generation’s EDR on the worst-fitness-value axis is 1.58 times that of the initial generation, with the SDR 1.76 times greater than that of the initial generation. On the best-fitness-value axis, the EDR is 2.19 times greater and the SDR is 2.47 times greater than for the initial generation. These results indicate that population diversity has improved in both dimensions, with a more significant improvement in the best-fitness-value dimension.

**Table 8.** Calculation of scenario population diversity metrics.

| | | **Primary Scenario Population** | | | | | | **Last Scenario Population** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Instance** | **Scale** | **Feature of Worst Fitness Value** | | | **Feature of Best Fitness Value** | | | **Feature of Worst Fitness Value** | | | **Feature of Best Fitness Value** | | |
| **Number** | **($n \times m_1 \times m_2 \times R$)** | **Max** | **Min** | $\sigma$ | **Max** | **Min** | $\sigma$ | **Max** | **Min** | $\sigma$ | **Max** | **Min** | $\sigma$ |
| 1 | $10 \times 2 \times 2 \times 0.6$ | 1529 | 1028 | 115 | 835 | 640 | 69 | 1553 | 1039 | 207 | 908 | 574 | 143 |
| 2 | $10 \times 2 \times 2 \times 1.2$ | 2725 | 2356 | 105 | 780 | 393 | 106 | 2746 | 2374 | 114 | 797 | 391 | 133 |
| 3 | $10 \times 3 \times 3 \times 0.6$ | 2657 | 1842 | 245 | 1126 | 743 | 113 | 2652 | 1770 | 237 | 1205 | 631 | 157 |
| 4 | $10 \times 3 \times 3 \times 1.2$ | 2564 | 1810 | 250 | 775 | 466 | 74 | 2636 | 1827 | 193 | 860 | 417 | 120 |
| 5 | $10 \times 4 \times 4 \times 0.6$ | 3297 | 2862 | 129 | 920 | 661 | 79 | 3277 | 2741 | 172 | 1015 | 573 | 144 |
| 6 | $10 \times 4 \times 4 \times 1.2$ | 2130 | 1836 | 72 | 596 | 268 | 138 | 2307 | 1992 | 87 | 699 | 222 | 164 |
| 7 | $15 \times 2 \times 2 \times 0.6$ | 2161 | 1814 | 91 | 1253 | 1058 | 100 | 2324 | 1855 | 151 | 1484 | 974 | 166 |
| 8 | $15 \times 2 \times 2 \times 1.2$ | 3843 | 3601 | 76 | 554 | 446 | 109 | 4238 | 3634 | 192 | 885 | 421 | 143 |
| 9 | $15 \times 3 \times 3 \times 0.6$ | 3216 | 2702 | 142 | 1458 | 995 | 108 | 3639 | 2502 | 343 | 1741 | 946 | 239 |
| 10 | $15 \times 3 \times 3 \times 1.2$ | 2419 | 2138 | 74 | 755 | 621 | 112 | 2424 | 2052 | 108 | 827 | 577 | 72 |
| 11 | $15 \times 4 \times 4 \times 0.6$ | 2520 | 2157 | 90 | 1211 | 873 | 205 | 2684 | 2144 | 175 | 1447 | 775 | 207 |
| 12 | $15 \times 4 \times 4 \times 1.2$ | 1455 | 1073 | 116 | 830 | 452 | 146 | 1786 | 1043 | 223 | 1406 | 377 | 338 |
| 13 | $20 \times 2 \times 2 \times 0.6$ | 4624 | 4069 | 164 | 1552 | 1421 | 90 | 5037 | 4019 | 284 | 1757 | 1334 | 134 |
| 14 | $20 \times 2 \times 2 \times 1.2$ | 1315 | 1026 | 78 | 796 | 505 | 100 | 1560 | 971 | 174 | 1105 | 475 | 206 |
| 15 | $20 \times 3 \times 3 \times 0.6$ | 3260 | 2789 | 148 | 1614 | 1340 | 69 | 3552 | 2702 | 249 | 1805 | 1223 | 135 |
| 16 | $20 \times 3 \times 3 \times 1.2$ | 4320 | 3623 | 185 | 1442 | 1025 | 106 | 4306 | 3556 | 192 | 1801 | 843 | 241 |
| 17 | $20 \times 4 \times 4 \times 0.6$ | 6807 | 5883 | 216 | 2018 | 1630 | 113 | 6976 | 5752 | 314 | 2376 | 1377 | 238 |
| 18 | $20 \times 4 \times 4 \times 1.2$ | 1992 | 1675 | 81 | 1006 | 740 | 74 | 2505 | 1686 | 240 | 1462 | 693 | 206 |
| 19 | $25 \times 5 \times 5 \times 0.6$ | 4598 | 3941 | 147 | 2436 | 2129 | 79 | 5023 | 3821 | 319 | 2853 | 1909 | 226 |
| 20 | $25 \times 5 \times 5 \times 1.2$ | 3975 | 3281 | 169 | 1675 | 1072 | 138 | 4118 | 3157 | 279 | 2194 | 854 | 393 |
| 21 | $25 \times 6 \times 6 \times 0.6$ | 3430 | 2737 | 149 | 1928 | 1442 | 100 | 3581 | 2475 | 289 | 2119 | 1241 | 234 |
| 22 | $25 \times 6 \times 6 \times 1.2$ | 7863 | 6426 | 323 | 1735 | 1204 | 109 | 8671 | 6289 | 748 | 2086 | 1048 | 288 |
| 23 | $30 \times 6 \times 6 \times 0.6$ | 7003 | 5792 | 286 | 3126 | 2657 | 108 | 7651 | 5357 | 614 | 3479 | 2295 | 329 |
| 24 | $30 \times 6 \times 6 \times 1.2$ | 4420 | 3592 | 174 | 1535 | 995 | 112 | 4814 | 3541 | 312 | 1892 | 938 | 249 |
| 25 | $30 \times 7 \times 7 \times 0.6$ | 7030 | 5722 | 342 | 3272 | 2470 | 205 | 7852 | 4774 | 798 | 3759 | 1945 | 479 |
| 26 | $30 \times 7 \times 7 \times 1.2$ | 5805 | 4666 | 312 | 1931 | 1353 | 146 | 5950 | 4457 | 345 | 2417 | 1202 | 303 |
| 27 | $35 \times 7 \times 7 \times 0.6$ | 4734 | 4092 | 117 | 2858 | 2502 | 90 | 4935 | 4031 | 283 | 3187 | 2468 | 222 |
| 28 | $35 \times 7 \times 7 \times 1.2$ | 5784 | 5128 | 168 | 1740 | 1277 | 100 | 5935 | 5096 | 242 | 2111 | 1071 | 275 |

**Table 9.** Calculation of the EDR and SDR on the two feature axes.

| Instance Number | Scale ($n \times m_1 \times m_2 \times R$) | Feature of Worst Fitness Value EDR | Feature of Worst Fitness Value SDR | Feature of Best Fitness Value EDR | Feature of Best Fitness Value SDR |
|---|---|---|---|---|---|
| 1 | $10 \times 2 \times 2 \times 0.6$ | 1.03 | 1.80 | 1.71 | 2.44 |
| 2 | $10 \times 2 \times 2 \times 1.2$ | 1.01 | 1.09 | 1.05 | 1.19 |
| 3 | $10 \times 3 \times 3 \times 0.6$ | 1.08 | 0.97 | 1.50 | 1.31 |
| 4 | $10 \times 3 \times 3 \times 1.2$ | 1.07 | 0.77 | 1.43 | 1.33 |
| 5 | $10 \times 4 \times 4 \times 0.6$ | 1.23 | 1.33 | 1.71 | 1.74 |
| 6 | $10 \times 4 \times 4 \times 1.2$ | 1.07 | 1.21 | 1.46 | 2.02 |
| 7 | $15 \times 2 \times 2 \times 0.6$ | 1.35 | 1.66 | 2.61 | 3.05 |
| 8 | $15 \times 2 \times 2 \times 1.2$ | 2.49 | 2.52 | 4.31 | 5.62 |
| 9 | $15 \times 3 \times 3 \times 0.6$ | 2.21 | 2.41 | 1.72 | 1.81 |
| 10 | $15 \times 3 \times 3 \times 1.2$ | 1.32 | 1.47 | 1.87 | 1.76 |
| 11 | $15 \times 4 \times 4 \times 0.6$ | 1.49 | 1.94 | 1.98 | 2.40 |
| 12 | $15 \times 4 \times 4 \times 1.2$ | 1.94 | 1.93 | 2.72 | 3.60 |
| 13 | $20 \times 2 \times 2 \times 0.6$ | 1.83 | 1.73 | 3.23 | 3.69 |
| 14 | $20 \times 2 \times 2 \times 1.2$ | 2.04 | 2.22 | 2.17 | 2.64 |
| 15 | $20 \times 3 \times 3 \times 0.6$ | 1.81 | 1.68 | 2.13 | 1.97 |
| 16 | $20 \times 3 \times 3 \times 1.2$ | 1.08 | 1.04 | 2.29 | 2.27 |
| 17 | $20 \times 4 \times 4 \times 0.6$ | 1.33 | 1.46 | 2.57 | 2.12 |
| 18 | $20 \times 4 \times 4 \times 1.2$ | 2.58 | 2.96 | 2.89 | 2.80 |
| 19 | $25 \times 5 \times 5 \times 0.6$ | 1.83 | 2.17 | 3.08 | 2.86 |
| 20 | $25 \times 5 \times 5 \times 1.2$ | 1.38 | 1.65 | 2.22 | 2.85 |
| 21 | $25 \times 6 \times 6 \times 0.6$ | 1.60 | 1.94 | 1.81 | 2.34 |
| 22 | $25 \times 6 \times 6 \times 1.2$ | 1.66 | 2.32 | 1.95 | 2.65 |
| 23 | $30 \times 6 \times 6 \times 0.6$ | 1.89 | 2.15 | 2.53 | 3.03 |
| 24 | $30 \times 6 \times 6 \times 1.2$ | 1.54 | 1.79 | 1.77 | 2.22 |
| 25 | $30 \times 7 \times 7 \times 0.6$ | 2.35 | 2.34 | 2.26 | 2.34 |
| 26 | $30 \times 7 \times 7 \times 1.2$ | 1.31 | 1.11 | 2.10 | 2.07 |
| 27 | $35 \times 7 \times 7 \times 0.6$ | 1.41 | 2.42 | 2.02 | 2.47 |
| 28 | $35 \times 7 \times 7 \times 1.2$ | 1.28 | 1.44 | 2.25 | 2.74 |
| Average | | 1.58 | 1.76 | 2.19 | 2.47 |

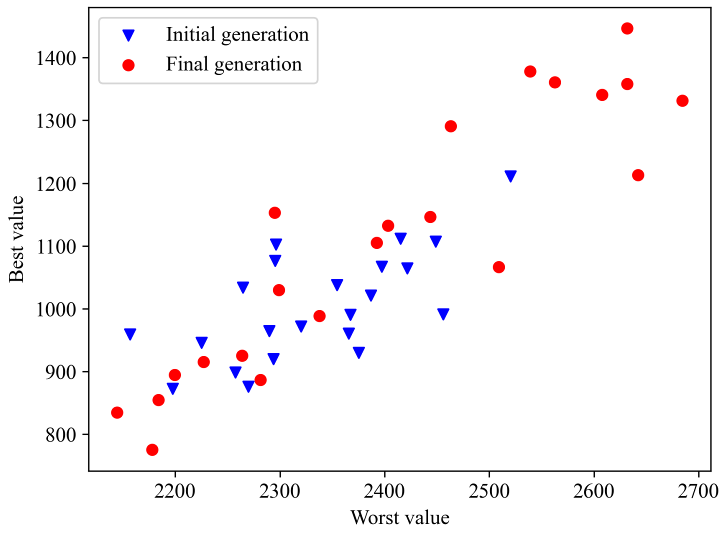


**Figure 14.** First-generation scenario populations vs. last-generation scenario populations (Instance 11).

### 5.4.2. Representativeness

Since the generated scenarios only have boundary information for processing times, it is difficult to assess the accuracy and representativeness of the generated scenarios. Nonetheless, we can measure them based on the precision. $A$ representative scenario population should have a similar impact on the solutions. To test this, we evaluated the best solution found for each data instance using both the final generation's $P$ scenario from the same run and $10P$ scenarios generated from ten runs. By calculating the fitness value increase percentage ($FVIP$), we compared the differences in fitness values for the best solutions from these two sets of scenarios, as shown in Equation (33), where $S$ represents the set of $P$ scenarios from the final generation of the same run, and $AS$ represents the set of $10P$ scenarios generated from ten runs.

$$FVIP = \frac{\sum_{s \in AS} F(x,s)/|AS| - \sum_{s \in S} F(x,s)/|S|}{\sum_{s \in S} F(x,s)/|S|} \tag{33}$$

Figure 16 shows the test results for 28 data instances. On average, the impact on fitness of the two scenario sets is similar, with a range of 0.2% to 8.7%. The table shows more negative values, indicating that the scenarios generated in the same run ($S$) tend to slightly underestimate the performance of the best solution. However, this is an average trend rather than a universal rule.

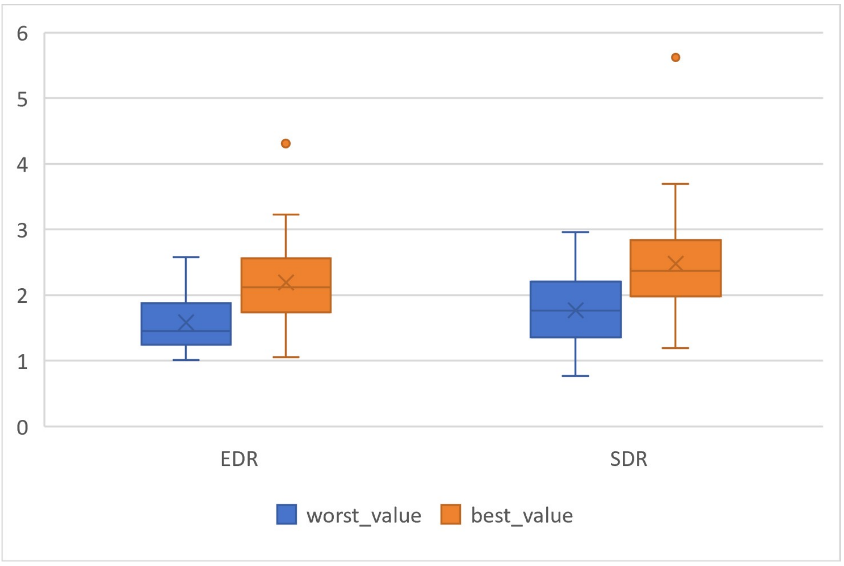


**Figure 15.** EDR and SDR on two feature axes.

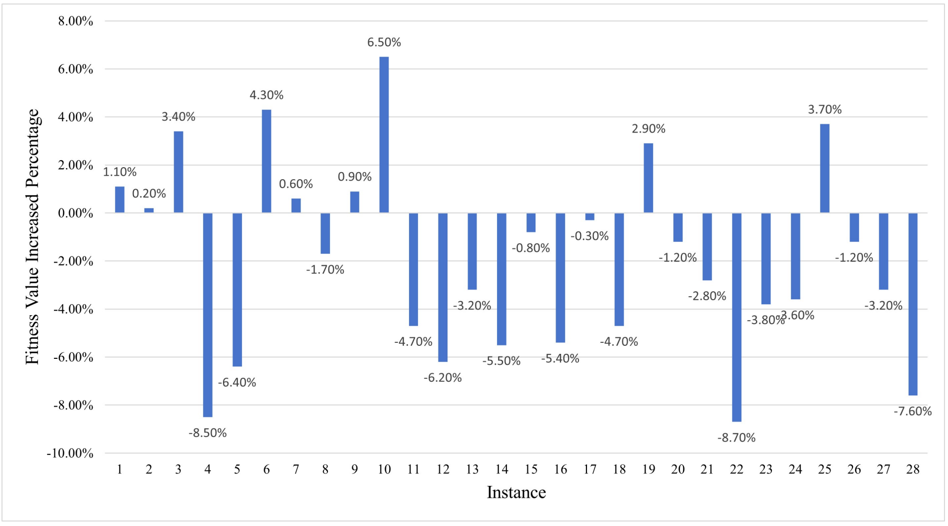


**Figure 16.** Fitness value increase percentage.

### 5.5. Decision Support

Ultimately, this method aims to support decision-makers by providing a range of high-quality solutions tailored to different risk preferences, supplemented by visual tools to demonstrate the potential impacts of uncertainty on these solutions. This section will delve into the outputs provided to decision-makers and the potential computational limitations.

The number of solutions obtainable using this method depends on the available time or computational resources. Even with limited resources, each solution's fitness criteria can be run once to compare the best solutions generated under two different evaluation metrics. For example, Figure 17 shows the best results for Instance 24 under different evaluation criteria. The vertical axis represents the objective value of each optimal solution in each scenario. In this data instance, the solution under the pessimistic criterion performs better in most scenarios, while the solution under the average criterion performs better in a few scenarios. Besides comparing the objective values of different solutions, the solutions can also be converted into Gantt charts to visually display scheduling and workshop suspensions (see Figures 18 and 19). This feature is crucial for applying the method in decision support systems. For ', decision-makers can adjust the Gantt chart to modify the final solution and test these new solutions within the generated scenario set. Additionally, decision-makers can include specific scenarios in the initial population. However, there must be certain limitations to ensure that the number of randomly generated individuals in the population meets the minimum requirements.

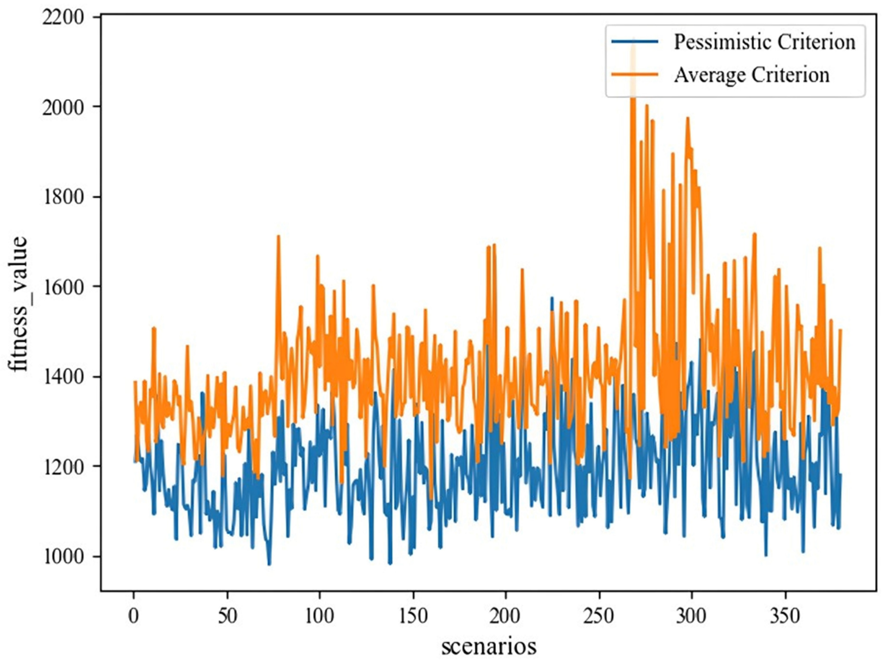


**Figure 17.** Final generated objective value for optimal solution (Instance 24).

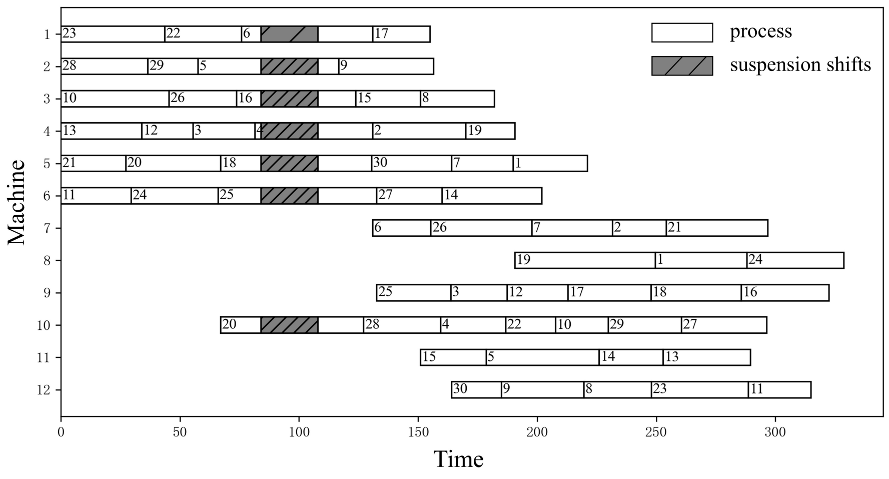


**Figure 18.** Gantt chart of optimal solutions under the average criterion (Instance 24).

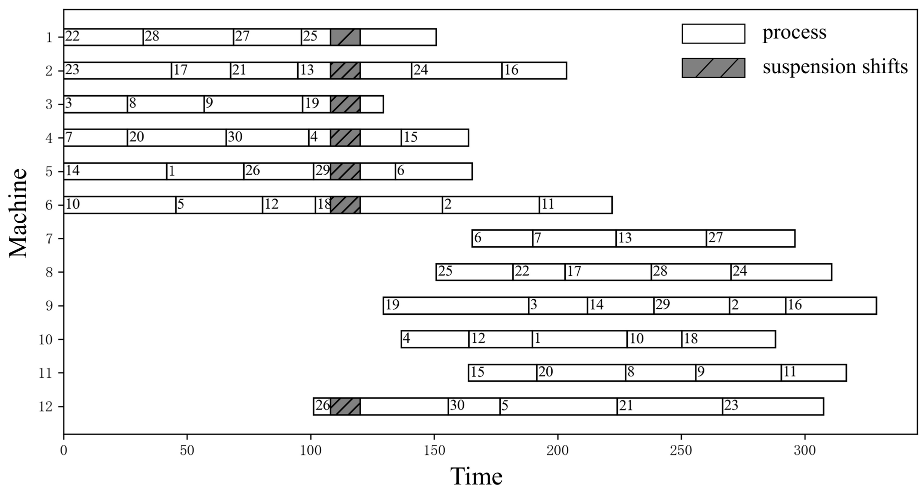


**Figure 19.** Gantt chart of optimal solutions under the pessimistic criterion (Instance 24).

## 6. Conclusions

This study regards workshop suspension shifts as a tool to adjust capacity and ensure on-time delivery. We propose a two-stage hybrid flow shop scheduling model with suspension shifts, addressing the gap in integrated the optimization of workshop suspensions and scheduling. We design the DCE-BRKGA to solve the scheduling problem under uncertain processing times. This method is highly applicable and provides robust solutions without requiring decision-makers to define scenarios or their probability distributions, needing only the upper and lower bounds of uncertainty parameters. In terms of application, the study considers both average and pessimistic criteria, offering a comprehensive information framework for decision-makers and enhancing adaptability and accuracy in uncertain environments through visual comparisons of the solutions. We quantified the benefits by using the VSS and EVPI on 28 datasets. Compared to the average scenario, the VSS results show that the proposed algorithm achieves additional value gains ranging from 0.9% to 69.9%. Furthermore, the EVPI indicates that the algorithm could potentially improve outcomes by 2.4% to 20.3% after eliminating uncertainty. These results demonstrate that the DCE-BRKGA effectively provides robust solutions even in the absence of known processing time distributions.

In this work, we assume that all workshop suspension shifts have the same priority. However, if we prioritize suspension shifts during rest days to reduce overtime, the model could have broader practical applications. Additionally, our current approach sets the objective as $Min \sum_{i \in N}(\alpha_i E_i + \beta_i T_i)$, effectively balancing the conflicting objectives of minimizing earliness and tardiness through a weighted sum. While this method addresses these objectives within a single optimization framework, it does not fully explore the potential of multi-objective optimization, particularly in terms of generating a Pareto frontier of optimal solutions. Future work could focus on developing algorithms that explicitly consider Pareto-optimal solutions, offering a more comprehensive analysis of the trade-offs between multiple objectives. Moreover, optimizing the algorithm to reduce computation time remains crucial. Currently, evaluating the fitness of all individuals results in high computational costs, whereas sampling too few individuals increases the algorithm's randomness. Therefore, effective sampling strategies and fitness evaluation are important areas for further exploration to enhance the algorithm's efficiency and robustness.

**Author Contributions:** Conceptualization, L.H. and D.L.; Data curation, L.H.; Formal analysis, Z.H. and L.H.; Funding acquisition, D.L.; Investigation, Z.H. and L.H.; Methodology, L.H. and D.L.; Project administration, D.L.; Resources, D.L.; Software, Z.H. and L.H.; Supervision, D.L.; Validation, Z.H. and D.L.; Visualization, Z.H. and L.H.; Writing—original draft, Z.H. and L.H.; Writing—review and editing, Z.H. and D.L. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research is supported by National Natural Science Foundation of China [grant No. 72171054] and the Fujian Provincial Natural Science Foundation [grant No. 2023J06015].

**Data Availability Statement:** The original data presented in the study are openly available in GitHub at https://github.com/nbzj/Data-Instances-of-Co-Evolutionary-Algorithm-for-THFSP-with-suspension-shifts.

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

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| DCE-BRKGA | dual-space co-evolutionary biased random key genetic algorithm |
| BRKGA | biased random key genetic algorithm |
| EVPI | expected value of perfect information |
| $EVPI_{\%}$ | expected value of perfect information as a percentage |
| VSS | value of the stochastic solution |
| $VSS_{\%}$ | value of the stochastic solution as a percentage |
| HFSP | hybrid flow shop scheduling problem |
| THFSP | two-stage hybrid flow shop scheduling problem |
| RKGA | random key genetic algorithm |
| RA | randomized adaptation |
| WS | wait-and-see |
| EEV | expected value using the expected solution |
| FVIP | fitness value increase percentage |
| EDR | extremum deviation ratio |
| SDR | standard deviation ratio |

## References

1. Meng, L.; Zhang, C.; Shao, X.; Zhang, B.; Ren, Y.; Lin, W. More MILP models for hybrid flow shop scheduling problem and its extended problems. *Int. J. Prod. Res.* **2020**, *58*, 3905–3930. [CrossRef]
2. Wang, S.; Liu, M. A genetic algorithm for two-stage no-wait hybrid flow shop scheduling problem. *Comput. Oper. Res.* **2013**, *40*, 1064–1075. [CrossRef]
3. Oliveira, B.B.; Carravilla, M.A.; Oliveira, J.F.; Costa, A.M. A co-evolutionary matheuristic for the car rental capacity-pricing stochastic problem. *Eur. J. Oper. Res.* **2019**, *276*, 637–655. [CrossRef]
4. Lin, R.; Wang, J.Q.; Oulamara, A. Online scheduling on parallel-batch machines with periodic availability constraints and job delivery. *Omega* **2023**, *116*, 102804. [CrossRef]
5. Nguyen, A.H.G.; Sheen, G.J.; Yeh, Y. An approximation algorithm for the two identical parallel machine problem under machine availability constraints. *J. Ind. Prod. Eng.* **2023**, *40*, 54–67. [CrossRef]
6. Yu, T.S.; Han, J.H. Scheduling proportionate flow shops with preventive machine maintenance. *Int. J. Prod. Econ.* **2021**, *231*, 107874. [CrossRef]
7. Nicosia, G.; Detti, P.; Pacifici, A. Robust Job-Sequencing with an Uncertain Flexible Maintenance Activity. *Comput. Ind. Eng.* **2023**, *185*, 109610.
8. Lee, C.Y.; Leon, V.J. Machine scheduling with a rate-modifying activity. *Eur. J. Oper. Res.* **2001**, *128*, 119–128. [CrossRef]
9. Nourelfath, M.; Châtelet, E. Integrating production, inventory and maintenance planning for a parallel system with dependent components. *Reliab. Eng. Syst. Saf.* **2012**, *101*, 59–66. [CrossRef]
10. Lu, Z.; Zhang, Y.; Han, X. Integrating run-based preventive maintenance into the capacitated lot sizing problem with reliability constraint. *Int. J. Prod. Res.* **2013**, *51*, 1379–1391. [CrossRef]
11. Liu, Y.; Zhang, Q.; Ouyang, Z.; Huang, H.Z. Integrated production planning and preventive maintenance scheduling for synchronized parallel machines. *Reliab. Eng. Syst. Saf.* **2021**, *215*, 107869. [CrossRef]
12. Zheng, X.; Zhou, S.; Xu, R.; Chen, H. Energy-efficient scheduling for multi-objective two-stage flow shop using a hybrid ant colony optimisation algorithm. *Int. J. Prod. Res.* **2020**, *58*, 4103–4120. [CrossRef]
13. Shahgholi Zadeh, M.; Katebi, Y.; Doniavi, A. A heuristic model for dynamic flexible job shop scheduling problem considering variable processing times. *Int. J. Prod. Res.* **2019**, *57*, 3020–3035. [CrossRef]
14. Framinan, J.M.; Fernandez-Viagas, V.; Perez-Gonzalez, P. Using real-time information to reschedule jobs in a flowshop with variable processing times. *Comput. Ind. Eng.* **2019**, *129*, 113–125. [CrossRef]
15. Yue, Q.; Zhou, S. Due-window assignment scheduling problem with stochastic processing times. *Eur. J. Oper. Res.* **2021**, *290*, 453–468. [CrossRef]

16. Ghaedy-Heidary, E.; Nejati, E.; Ghasemi, A.; Torabi, S.A. A simulation optimization framework to solve stochastic flexible job-shop scheduling problems—Case: Semiconductor manufacturing. *Comput. Oper. Res.* **2024**, *163*, 106508. [CrossRef]
17. Liu, X.; Chu, F.; Zheng, F.; Chu, C.; Liu, M. Parallel machine scheduling with stochastic release times and processing times. *Int. J. Prod. Res.* **2021**, *59*, 6327–6346. [CrossRef]
18. Lu, C.C.; Ying, K.C.; Lin, S.W. Robust single machine scheduling for minimizing total flow time in the presence of uncertain processing times. *Comput. Ind. Eng.* **2014**, *74*, 102–110. [CrossRef]
19. Wang, S.; Cui, W. Approximation algorithms for the min-max regret identical parallel machine scheduling problem with outsourcing and uncertain processing time. *Int. J. Prod. Res.* **2021**, *59*, 4579–4592. [CrossRef]
20. Xiao, S.; Wu, Z.; Dui, H. Resilience-Based Surrogate Robustness Measure and Optimization Method for Robust Job-Shop Scheduling. *Mathematics* **2022**, *10*, 4048. [CrossRef]
21. Ali, O.; Abbas, Q.; Mahmood, K.; Bautista Thompson, E.; Arambarri, J.; Ashraf, I. Competitive Coevolution-Based Improved Phasor Particle Swarm Optimization Algorithm for Solving Continuous Problems. *Mathematics* **2023**, *11*, 4406. [CrossRef]
22. Lei, H.; Wang, R.; Laporte, G. Solving a multi-objective dynamic stochastic districting and routing problem with a co-evolutionary algorithm. *Comput. Oper. Res.* **2016**, *67*, 12–24. [CrossRef]
23. Zhao, F.; He, X.; Wang, L. A two-stage cooperative evolutionary algorithm with problem-specific knowledge for energy-efficient scheduling of no-wait flow-shop problem. *IEEE Trans. Cybern.* **2020**, *51*, 5291–5303. [CrossRef] [PubMed]
24. Xiao, Q.z.; Zhong, J.; Feng, L.; Luo, L.; Lv, J. A cooperative coevolution hyper-heuristic framework for workflow scheduling problem. *IEEE Trans. Serv. Comput.* **2019**, *15*, 150–163. [CrossRef]
25. Wang, Z.Y.; Pan, Q.K.; Gao, L.; Jing, X.L.; Sun, Q. A cooperative iterated greedy algorithm for the distributed flowshop group robust scheduling problem with uncertain processing times. *Swarm Evol. Comput.* **2023**, *79*, 101320. [CrossRef]
26. Ming, F.; Gong, W.; Wang, L.; Lu, C. A tri-population based co-evolutionary framework for constrained multi-objective optimization problems. *Swarm Evol. Comput.* **2022**, *70*, 101055. [CrossRef]
27. Gu, J.; Gu, M.; Cao, C.; Gu, X. A novel competitive co-evolutionary quantum genetic algorithm for stochastic job shop scheduling problem. *Comput. Oper. Res.* **2010**, *37*, 927–937. [CrossRef]
28. Herrmann, J.W. A genetic algorithm for minimax optimization problems. In Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), Washington, DC, USA, 6–9 July 1999; Volume 2, pp. 1099–1103.
29. Resende, M.G.; de Sousa, J.P.; Jensen, M.T. A new look at solving minimax problems with coevolutionary genetic algorithms. In *Metaheuristics Computer Decision-Making*; Springer: Boston, MA, USA, 2004; pp. 369–384.
30. Jensen, M.T. *Robust and Flexible Scheduling with Evolutionary Computation*; Citeseer: Princeton, NJ, USA, 2001.
31. He, X.; Pan, Q.K.; Gao, L.; Wang, L.; Suganthan, P.N. A greedy cooperative co-evolutionary algorithm with problem-specific knowledge for multiobjective flowshop group scheduling problems. *IEEE Trans. Evol. Comput.* **2021**, *27*, 430–444. [CrossRef]
32. Suhaimi, N.; Nguyen, C.; Damodaran, P. Lagrangian approach to minimize makespan of non-identical parallel batch processing machines. *Comput. Ind. Eng.* **2016**, *101*, 295–302. [CrossRef]