

## Article

# An Adaptive Co-Evolutionary Memetic Algorithm for a Hybrid Flow Shop Scheduling Problem with Sequence-Dependent Setup and Transportation Times

Dekun Wang <sup>1</sup>, Yu Lei <sup>1</sup>, Zhengang Yuan <sup>1</sup>, Yuhao Zhao <sup>1</sup>, Yubin Wang <sup>1</sup>, Wenjie Wang <sup>1,\*</sup>  and Gang Yuan <sup>2</sup>

<sup>1</sup> School of Mechanical and Power Engineering, Zhengzhou University, Zhengzhou 450001, China; dekunwang@gs.zzu.edu.cn (D.W.); ly202441020105@stu.zzu.edu.cn (Y.L.); yuanzg@gs.zzu.edu.cn (Z.Y.); zyh000729@gs.zzu.edu.cn (Y.Z.); wangyubin@stu.zzu.edu.cn (Y.W.)

<sup>2</sup> College Engineering, Nanjing Agricultural University, Nanjing 210095, China; 2025092@njau.edu.cn

\* Correspondence: wenjiewang@zzu.edu.cn

## Abstract

The hybrid flow shop scheduling problem (HFSP) with unrelated parallel machines (UPMs), sequence-dependent setup times (SDSTs), and inter-stage transportation times has recently emerged as a prominent research topic. To address this scheduling problem with the objective of minimizing the maximum completion time (makespan), this paper first formulates a mixed-integer linear programming (MILP) model based on the machine-position modeling idea. Exact solution analyses on small-scale instances reveal that the strong coupling effect of these triple constraints concentrates the computational bottleneck on the time-consuming proof of optimality, thereby underscoring the strongly NP-hard nature of the investigated HFSP-SDST-T problem. To efficiently solve large-scale instances, a novel adaptive co-evolutionary memetic algorithm (ACMA) is proposed. ACMA adopts a dual-population co-evolutionary framework, where a customized genetic algorithm (GA) is designed for global exploration and a Lévy flight-enhanced particle swarm optimization (PSO) improves local search capability. To dynamically balance exploration and exploitation, a Dynamic Role Allocation (DRA) mechanism is developed to adaptively reassign individuals between the two populations according to their evolutionary states. Moreover, a progressive two-stage memetic enhancement strategy is proposed to overcome premature convergence by sequentially activating deep variable neighborhood search (VNS) and a catastrophe-based diversification strategy, enabling adaptive responses to different stagnation levels. Extensive experiments, including ablation studies, comparisons with benchmark algorithms, and computational complexity analysis, are conducted on small- and large-scale instances. The results show that ACMA consistently obtains the exact optimal solutions obtained from the MILP model for small-scale instances and achieves competitive performance on large-scale complex instances. Furthermore, Wilcoxon signed-rank tests confirm the statistical significance of the performance differences, supporting the reliability of the experimental results.

**Keywords:** hybrid flow shop scheduling; SDST; co-evolutionary memetic algorithms; UPM; mixed-integer linear programming



Academic Editors: Dan Zhang and Angelos P. Markopoulos

Received: 7 June 2026

Revised: 17 July 2026

Accepted: 25 August 2026

Published: 27 August 2026

**Copyright:** © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

## 1. Introduction

### 1.1. Background and Motivation

The hybrid flow shop scheduling problem (HFSP) extends the classical flow shop configuration by incorporating multiple parallel machines at bottleneck stages to balance

machine workloads and enhance production capacity. Due to its extensive applications in modern industries—such as textiles, pharmaceuticals, semiconductor manufacturing, automobile assembly, and steel production—it has remained a focal point of research in scheduling. Given its proven NP-hard complexity, the development of efficient intelligent optimization algorithms to obtain high-quality approximate solutions has become a mainstream consensus in both academia and industry. Recent research trends have shifted from solving idealized classical HFSP models toward extended models that integrate more realistic constraints, such as sequence-dependent setup times (SDSTs), transportation times, unrelated parallel machines (UPMs), and sustainability factors (e.g., worker fatigue and energy consumption). This evolution ensures that scheduling models more closely align with complex, real-world production environments.

## 1.2. Literature Review

Given its practical relevance in modern manufacturing systems, the hybrid flow shop scheduling problem (HFSP) has attracted extensive research attention due to its inherent complexity and computational challenges. Developing efficient intelligent optimization algorithms to obtain high-quality solutions within reasonable computational time has therefore become a critical research topic. In recent years, research interest has gradually extended from idealized classical HFSP models toward more realistic scheduling environments involving various production constraints, aiming to better represent the complexity of real-world manufacturing processes. Accordingly, numerous studies have investigated advanced solution approaches for HFSP variants, including improved encoding and decoding schemes, swarm intelligence-based optimization strategies, and effective local search operators, particularly for problems involving unrelated parallel machines (UPMs), sequence-dependent setup times (SDSTs), and transportation constraints.

### 1.2.1. Related Studies on Classical HFSP

HFSP, an extension of parallel machine and flow shop scheduling, presents a core challenge: the deep coupling between job sequencing and machine assignment decisions. Following the framework established by Yu [1] and related literature, the evolutionary trajectory of HFSP research can be characterized by the diversification of application scenarios and refinement of optimization strategies.

Based on the dimensionality of the optimization objectives, existing studies can be broadly categorized into single- and multi-objective HFSP. In single-objective research, minimizing the maximum completion time (makespan) is the most direct and prevalent metric for evaluating production efficiency. Tang et al. [2] thoroughly investigated this objective within a distributed manufacturing environment using an improved Cuckoo Search algorithm. Zhang et al. [3] addressed a flexible flow shop scheduling problem with limited buffers and two process routings, proposing a discrete whale optimization algorithm. Furthermore, Han et al. [4] developed an improved migrating bird optimization algorithm to solve the HFSP; their approach employed a simulated annealing-based acceptance criterion to maintain population diversity and incorporated a competitive mechanism to enhance the exploration capability of the algorithm. In multi-objective research, alongside traditional production efficiency metrics, factors such as energy conservation, emission reduction, due-date deviations, and processing costs have garnered increasing attention. For instance, Yue et al. [5] formulated a multi-objective optimization model for a two-stage HFSP in printed circuit board manufacturing, balancing energy efficiency and weighted tardiness (WT). In an Internet of Things (IoT) environment, Dai et al. [6] constructed a multistage HFSP architecture and designed an improved estimation of distribution algorithm. Additionally, Hasani et al. [7] applied a non-dominated sorting genetic algorithm

to solve a flexible flow shop problem, aiming to strike a trade-off between production costs and energy consumption. The superiority of their algorithm was validated through experimental comparisons with SPEA2 and NSGA-II.

At the algorithmic design level, solution strategies for the HFSP have transitioned from single metaheuristics to hybrid enhancement strategies. Optimization efforts have predominantly focused on local search methods, swarm intelligence, and multi-population metaheuristic algorithms that hybridize these paradigms. Algorithms such as the genetic algorithm (GA), particle swarm optimization (PSO) [8], whale optimization algorithm (WOA) [9], and artificial bee colony (ABC) [10] have demonstrated formidable global exploration capabilities in solving the classical HFSP. To further balance global exploration and local exploitation, researchers have incorporated intensive search mechanisms. Xu et al. [11] enhanced individual quality by integrating the NEH heuristic for population initialization, combined with a local search and an adaptive GA. Song et al. [12] embedded three neighborhood structures into an ant colony optimization algorithm to facilitate deep exploitation. With the rapid advancement of artificial intelligence, learning-based algorithms utilizing agent-based systems and machine learning [13] are being progressively applied to the HFSP.

#### 1.2.2. Related Studies on Shop Scheduling Issues Considering SDST

In practical production, setup operations, such as workpiece clamping, tool adjustments, and mold replacements, are frequently required. The duration of these operations is often dependent on the processing sequence of consecutive workpieces, which is referred to as SDST. Ignoring SDST considerations may lead to considerable deviations between scheduling results and actual production conditions, causing issues such as machine idle time and prolonged manufacturing cycles. Consequently, SDST-aware scheduling has attracted considerable research attention in recent years.

In the domain of flow shop scheduling, researchers have conducted in-depth explorations of algorithmic paradigm innovation and refined problem definitions. Building on the importance of accounting for SDST in production systems and reflecting the increasing complexity of scheduling problems, scholars have progressively advanced algorithmic approaches. Driven by the rapid advancement of metaheuristic algorithms, An et al. [14] investigated a two-machine flow shop scheduling problem in which the setup time on the second machine was sequence-dependent. They developed a heuristic algorithm and validated its effectiveness for instances with fewer than 30 jobs through random testing. Khare and Agrawal [15] pioneered the application of emerging population-based intelligent algorithms, including the squirrel search algorithm (SSA), WOA, and grey wolf optimizer (GWO), to the HFSP-SDST problem, broadening the boundaries of solution methodologies. Building upon these efforts, Ozsoydan and Sagir [16] proposed a learning-based iterated greedy search algorithm, which further enhanced the performance in minimizing the makespan by incorporating a learning mechanism. The deepening of research is evident not only at the algorithmic level but also in the meticulous characterization of the problem. For instance, Liao et al. [17] provided a thorough analysis of two distinct types of SDST: one dependent solely on the job sequence and the other dependent on both the job sequence and machine assignment. This distinction aligns models more closely with complex production realities. Furthermore, a recent study by Jemmali and Hidri [18] proposed a GA-based metaheuristic and concurrently derived three lower bounds for the problem, establishing an important theoretical benchmark for the performance evaluation of subsequent algorithms in this field. In the context of job shop scheduling, Shen et al. [19] utilized a tabu search (TS) framework to develop a specialized neighborhood search and algorithmic structure, effectively solving the flexible job shop scheduling problem with

an SDST. Bektu and Sarac [20] extended the scope of the problem by proposing a mixed-integer linear programming (MILP) model and applying TS and SA to solve the unrelated parallel machine scheduling problem, considering both sequence-dependent setup times and machine eligibility constraints. Finally, Kim et al. [21] presented a novel mathematical model tailored for metaheuristics to address the identical parallel machine scheduling problem with SDST and proposed two encoding schemes and three decoding methods for extracting feasible schedules from metaheuristic solutions.

### 1.2.3. Scheduling Research Considering Transportation Times

Traditional HFSP models often assume negligible transportation delays between upstream and downstream operations. However, in practical manufacturing systems, the physical transfer of workpieces between processing stages inevitably introduces transportation time. Such delays may affect machine utilization and increase idle time, resulting in deviations between theoretical schedules and actual production operations. Consequently, increasing attention has been devoted to transportation-aware HFSP scheduling to improve the applicability of scheduling solutions in real-world manufacturing environments.

Regarding exact solutions and mathematical modeling, the introduction of transportation times fundamentally reshapes the evaluation models for makespan. In the study of multistage flow shop group scheduling, Liou and Hsieh [22] redefined machine idle time resulting from inter-stage transportation through rigorous mathematical expressions, confirming the strong coupling constraints between upstream and downstream logistics and operations. Furthermore, for the two-stage HFSP with transportation times, Hidri [23] developed a branch-and-bound framework that prunes the search tree by leveraging the structural symmetry of the problem and heuristic rules.

At the level of heuristic and intelligent optimization algorithms, metaheuristic strategies have become mainstream owing to the computational bottlenecks of exact algorithms. Xuan et al. [24] conducted an in-depth exploration of transportation scheduling in dynamic environments, emphasizing the critical role of logistical coordination in enhancing the efficiency of flexible flow lines. Liou et al. [22] investigated a class of flow shop group scheduling problems incorporating both SDST and inter-stage transportation times, proposing a hybrid PSO and GA while providing four theoretical lower bounds for the optimal solution. Hidri and Tlija [25] proposed a two-stage heuristic algorithm that constructed feasible solutions and optimizes bottleneck machines alternately to balance processing and logistics times. Finally, Yuan et al. [26] addressed a two-stage flow shop group scheduling problem with bidirectional transportation time constraints, formulating an MILP model to minimize the makespan and designing an improved iterated greedy algorithm tailored to the problem characteristics.

### 1.2.4. Analysis of UPMs in HFSP

Hybrid flow shop architectures incorporate parallel machine units to balance workloads across stages and increase overall system capacity. The academic community classifies parallel machine environments into three categories based on machine characteristics and job interaction logic: Identical Parallel Machines (IPMs), Heterogeneous Parallel Machines (HPMs), and UPMs.

In the IPM environment, the processing time of a job and the setup time between consecutive jobs are identical across all available machines [27]. In the HPM environment, machines process the same job at proportional but differing speeds, and the setup times between consecutive jobs vary across different machines [28]. In the UPM environment, each stage contains at least one eligible machine, with at least one stage featuring parallel units.

Crucially, the processing times of the same job vary across machines and are independent of each other.

The variations in parallel machine configurations directly dictate the complexity of scheduling decisions. In an IPM environment, because the processing and setup times for a given job are uniform across all machines in the same stage, the decision-making focus lies entirely on job sequencing. Conversely, in a HPM environment, although machines exhibit proportional speed differences when processing the same job, this performance disparity is globally consistent. In contrast, the UPM environment is both highly realistic and profoundly challenging: job processing times are mutually independent, depending not only on the inherent attributes of the operations but also on the specific job-machine matching suitability [29]. This profound heterogeneity necessitates that the scheduling task simultaneously addresses two orthogonal decision dimensions: determining the optimal machine assignment for each operation and establishing the optimal job sequencing for each assigned machine.

Since processing times in the UPM environment exhibit discrete and non-proportional characteristics, the decision space expands exponentially from one-dimensional sequencing to two-dimensional combinatorial optimization, encompassing both job sequencing and machine assignment. Consequently, traditional greedy algorithms are highly susceptible to becoming trapped in local optima. To overcome this, researchers have pursued multidimensional innovations in encoding/decoding schemes and search operators. At the decoding level, Song et al. [12] proposed five decoding rules tailored to UPM characteristics to capture optimal job-machine matching relationships through diversified mapping mechanisms. Furthermore, noting the impact of the initial solution quality on the evolutionary efficiency in UPM scenarios, Xu et al. [11] constructed a high-quality initial population for an adaptive GA by integrating the NEH heuristic. Additionally, to address the representation challenges of UPM assignment decisions, Xuan et al. [24] proposed a cellular matrix encoding scheme based on batch jobs, enabling the efficient modeling of complex machine configurations.

#### 1.2.5. Critical Analysis of Existing Methods

Building on prior work, substantial progress has been made in solving various HFSP variants, yet several critical challenges remain unaddressed.

From a modeling perspective, realistic production constraints have been increasingly incorporated into HFSP formulations. For SDST, Bektu and Sarac [20] developed an MILP model and solved it via TS and SA. For transportation-constrained problems, studies [22,23] further accounted for inter-stage logistics impacts. Other works focus on UPM resource allocation. Nevertheless, most existing studies address only one or a subset of these factors, and the strong coupling effects among UPM, SDST, and transportation times remain underexplored.

From an algorithmic standpoint, swarm intelligence and hybrid memetic algorithms are mainstream approaches for complex scheduling problems. Classic methods (e.g., PSO, ABC) converge quickly but suffer from diversity degradation and premature convergence in complex combinatorial spaces. Hybrid memetic algorithms, which integrate complementary search mechanisms, have been developed to boost overall performance. However, most such frameworks adopt predefined collaboration schemes: component roles and individual-component assignments are fixed at design time and static throughout optimization. For instance, the algorithm proposed in [30] leverages multiple subpopulations for better HFSP performance, but inter-subpopulation collaboration follows a preset structure, with no dynamic reassignment of individuals based on real-time search states.

Furthermore, most memetic algorithms trigger local search at fixed intervals or via predefined rules. For example, the work in [31] combines PSO with a learning-based

local search to enhance exploitation, but search execution follows rigid preset rules and cannot adapt intervention intensity to the degree of population stagnation. While fixed triggering improves local refinement, it may incur unnecessary computational overhead during normal convergence stages.

### 1.3. Main Work and Contributions

Compared with existing research, the main contributions of this study are summarized as follows:

- (1) This study investigates a HFSP that integrates the interactions between UPM, SDST, and inter-stage transportation times (shortly denoted as HFSP-SDST-T).
- (2) An accurate benchmark model for the HFSP-SDST-T is constructed based on the machine-position modeling approach, enabling precise representation of the three integrated constraints and an exact solution for small-scale instances.
- (3) A novel Adaptive Co-evolutionary Memetic Algorithm (ACMA) is proposed, incorporating a Dynamic Role Allocation (DRA) mechanism and a responsive two-stage enhancement engine to efficiently solve this large-scale problem.
- (4) The accuracy of the MILP model and the superiority of ACMA are validated through a series of test instances across different scales. Furthermore, the algorithm's internal mechanisms are analyzed to reveal how they effectively address the strong-coupling effects of deep-level constraints.

The remainder of this paper is organized as follows. Section 2 describes the HFSP-SDST-T problem and presents the MILP model. Section 3 provides a detailed introduction to the proposed ACMA and its core mechanism. Section 4 presents numerical experiments and provides an in-depth analysis of the algorithm's overall performance and the effectiveness of its core components. Finally, Section 5 concludes the paper and suggests directions for future work.

## 2. Problem Description and Mathematical Formulation

This section formally describes the investigated HFSP-SDST-T. Next, it defines comprehensive mathematical notations. Using these definitions, an MILP based on the machine-position concept is formulated to provide an exact benchmark for algorithm design and small-scale instance validation.

### 2.1. Problem Description and Constraint Analysis

The HFSP-SDST-T problem is described as follows: A set of  $n$  independent jobs  $J = \{1, 2, \dots, n\}$  must pass sequentially through  $j$  processing stages  $S = \{1, 2, \dots, j\}$  following identical process routings. At each stage  $s \in S$ , there is a set of  $M_s$  unrelated parallel machines, characterized by the fact that the processing time of the same job varies across different machines.

The problem considered in this study involves two complex constraints that are of critical practical importance in real-world manufacturing systems:

- (a) SDST: Once the processing of one job is completed on any machine, a setup (or changeover) operation must be performed before another different job can be processed. The duration of this setup operation is sequence-dependent, that is, it depends on the specific combination of the job just completed and the job to be processed next. In addition, this study assumes that the sequence-dependent setup operation can be completed in advance using the machine. Specifically, during the period in which a job is transported from the previous stage, the machine at the next stage is allowed to perform the required setup in advance. Accordingly, the actual start time of a job at the next stage must satisfy two conditions: it cannot be earlier than the

job's arrival time at that stage, and it cannot be earlier than the completion time of the corresponding setup operation on the assigned machine.

- (b) **Transportation Times:** After a job is completed on a machine at a specific stage, it must be transported to its designated machine at the next stage. This transportation process consumes a non-negligible amount of time, the duration of which depends on the origin and destination machines.

Therefore, the core scheduling task is to determine the processing sequence of jobs at each stage and the machine assignment for each job with the objective of minimizing the makespan. The following fundamental assumptions are adopted to ensure the model's rigor.

- (1). All jobs and machines are available at time zero.
- (2). Each job can be processed only once at each stage.
- (3). At any given time, a machine can process at most one job, and a job can be processed by at most one machine.
- (4). Once an operation is initiated, preemption or interruption is strictly prohibited.
- (5). The buffer capacity between stages is considered infinite; a job can commence transportation immediately after completion of its current processing stage.

## 2.2. Definition of Notations

The following section defines the sets, parameters, and decision variables. These elements are essential for rigorously building the mathematical model.

- $n$ : the number of jobs;
- $m$ : the number of machines;
- $j$ : the number of stages;
- $m_s$ : the number of machines at stage  $s$ ;
- $J$ : the set of jobs,  $J = \{1, 2, \dots, n\}$ ;
- $S$ : the set of stages,  $S = \{1, 2, \dots, j\}$ ;
- $M_s$ : the set of parallel machines at stages  $s$ ,  $M_s = \{1, 2, \dots, m_s\}$ ;
- $L$ : the set of processing positions on a single machine,  $L = \{1, 2, \dots, n\}$ ;
- $i, i'$ : the index of jobs,  $i, i' \in J (i \neq i')$ ;
- $s$ : the index of processing stages,  $s \in S$ ;
- $k$ : the index of machines,  $k \in M_s$ ;
- $q$ : the index of processing positions,  $q \in L$ ;
- $p_{i,s,k}$ : processing time of jobs  $i$  on machine  $k$  at stage  $s$ ;
- $SU_{i|i',i,s,k}$ : sequence-dependent setup time required on machine  $k$  at stage  $s$  to prepare for job  $i$  after completing job  $i'$ ;  $i|' = 0$  denotes the initial setup;
- $t_{k,k',s}$ : transportation time required to move a job from machine  $k$  at stage  $s$  to machine  $k'$  at stage  $s + 1$ ;
- $M$ : a sufficiently large positive number (Big- $M$ );
- $y_{i,s,k,q}$ : binary variable; 1 if job  $i$  is processed at position  $q$  on machine  $k$  at stage  $s$ , and 0 otherwise;
- $b_{i,s}$ : starting time of job  $i$  at stage  $s$ ;
- $e_{i,s}$ : completion time of job  $i$  at stage  $s$ ;
- $B_{s,k,q}$ : starting time of the operation at position  $q$  on machine  $k$  at stage  $s$ ;
- $E_{s,k,q}$ : completion time of the operation at position  $q$  on machine  $k$  at stage  $s$ ;
- $C_{\max}$ : maximum completion time (makespan).

## 2.3. Mathematical Formulation

To accurately characterize the problem and provide a precise benchmark for evaluating subsequent algorithms, this section develops an extended machine-position-based hybrid

integer-linearization model. The model follows the machine-position modeling paradigm originally proposed by Fattahi et al. [32] and is extended by incorporating sequence-dependent setup times and inter-stage transportation times.

This paper adopts a machine-position-based modeling approach rather than traditional adjacent-sequence-based or time-based models. A systematic comparative study by Wang [33,34] on the complexity of MILP scheduling models indicates that different modeling approaches exhibit significant differences in variable scale and relaxation compactness. Experimental results show that the number of Boolean decision variables in time-based models is closely related to the time upper bound. Since the introduction of SDST and transportation time in the problem addressed in this paper has significantly increased the maximum completion time, adopting a time-based modeling approach would inevitably lead to a variable-dimension explosion. Furthermore, compared to sequential models that require many decision variables to establish pairwise workpiece precedence relationships, machine-position-based modeling significantly reduces the total number of Boolean variables by allocating a finite set of slots, thereby substantially improving psolution efficiency.

Optimization objective:

$$\text{Minimize } C_{\max} \quad (1)$$

Constraints:

$$C_{\max} \geq e_{i,j} \forall i \in J \quad (2)$$

$$\sum_{k \in M_s} \sum_{q \in L} y_{i,s,k,q} = 1 \forall i \in J, s \in S \quad (3)$$

$$\sum_{i \in J} y_{i,s,k,q} \leq 1 \forall s \in S, k \in M_s, q \in L \quad (4)$$

$$\sum_{i \in J} y_{i,s,k,q} \leq \sum_{i \in J} y_{i,s,k,q-1} \forall s \in S, k \in M_s, q \in L / \{1\} \quad (5)$$

$$e_{i,s} = b_{i,s} + \sum_{k \in M_s} \sum_{q \in L} (p_{i,s,k} \cdot y_{i,s,k,q}) \forall i \in J, s \in S \quad (6)$$

$$E_{s,k,q} = B_{s,k,q} + \sum_{i \in J} (y_{i,s,k,q} \cdot p_{i,s,k}) \forall s \in S, k \in M_s, q \in L \quad (7)$$

$$b_{i,s} \geq B_{s,k,q} - M(1 - y_{i,s,k,q}) \forall i \in J, s \in S, k \in M_s, q \in L \quad (8)$$

$$b_{i,s} \leq B_{s,k,q} + M(1 - y_{i,s,k,q}) \forall i \in J, s \in S, k \in M_s, q \in L \quad (9)$$

$$B_{s,k,1} \geq \sum_{i \in J} (y_{i,s,k,1} \cdot SU_{0,i,s,k}) \forall s \in S, k \in M_s \quad (10)$$

$$B_{s,k,q} \geq E_{s,k,q-1} + SU_{i',i,s,k} - M(2 - y_{i,s,k,q} - y_{i',s,k,q-1}) \quad (11)$$

$$\forall i, i' \in J, s \in S, k \in M_s, q \in L / \{1\}$$

$$b_{i,s} \geq e_{i,s-1} + t_{k,k',s-1} - M(2 - \sum_{q \in L} y_{i,s-1,k,q} - \sum_{q \in L} y_{i,s,k',q}) \quad (12)$$

$$\forall i \in J, s \in S / \{1\}, k \in M_{s-1}, k' \in M_s$$

$$b_{i,s}, e_{i,s}, B_{s,k,q}, E_{s,k,q}, C_{\max} \geq 0 \quad (13)$$

$$y_{i,s,k,q} \in \{0, 1\} \quad (14)$$

Equations (1)–(2) define the objective of the model, which is to minimize the maximum completion time (makespan), denoted as  $C_{\max}$ .

Equations (3)–(5) formulate the job assignment constraints. Specifically, Equation (3) ensures that each job is assigned to exactly one machine and one specific position at each stage. Equation (4) guarantees that each processing position on any machine can

accommodate at most one job. Equation (5) enforces that jobs must be sequentially assigned to available positions on a machine without leaving any prior positions idle.

Equations (6)–(9) define the processing and time-linkage constraints. Equations (6) and (7) calculate the completion times, thereby ensuring the non-preemptive nature of the operations. By utilizing the Big-M method, Equations (8) and (9) establish a strict logical linkage between the starting time of a job ( $b_{i,s}$ ) and the starting time of its assigned machine position ( $B_{s,k,q}$ ). In the formula,  $B_{s,k,q}$  represents the machine's available time at position  $q$ . Sequence-dependent setup time may be completed in advance, and the job start time is taken as the later of the machine's setup completion time and the job arrival time.

Equations (10)–(12) represent the core time-related constraints. Equations (10) and (11) jointly handle the sequence-dependent setup times; the former defines the initial setup time for the first job assigned to a machine, while the latter determines the setup time between any two consecutive jobs. Equation (12) formulates the inter-stage transportation time constraint, ensuring that the processing of a job at the subsequent stage cannot commence until it has safely arrived from the preceding stage.

Finally, Equations (13) and (14) define the domains of all decision variables within the MILP model.

#### 2.4. An Illustrative Instance

To verify the correctness of the MILP model formulated in the previous section and intuitively demonstrate the inherent complexity of the investigated problem under study, a small-scale instance is presented. This instance comprises  $n = 5$  jobs and  $j = 2$  stages. Both stages are equipped with UPM, specifically  $m_1 = 2$  and  $m_2 = 2$ . Detailed data regarding the processing times, SDST, and inter-stage transportation times are provided in the Supplementary Data file.

The parameters of this instance are entered into the CPLEX solver (12.6.3.0) using the model formulated in Section 2.3. In a standard computational environment (Intel Core i5-13400F, 32 GB RAM Manufacturer: Intel Corporation, Santa Clara, California, USA.), after 261 s of computation, the solver ultimately finds the optimal solution, yielding a minimized makespan of 41.

Notably, the solving process of this instance (as depicted in Figure 1) revealed a critical phenomenon. As shown in Figure 1, it can be found that CPLEX has high efficiency in discovering solutions. The green line representing the 'best integer solution' (the incumbent) rapidly converges to the optimal value of 41 at a very early stage of the computation (likely within the first few dozen seconds). This indicates that discovering a high-quality or optimal feasible solution is not particularly difficult for CPLEX.

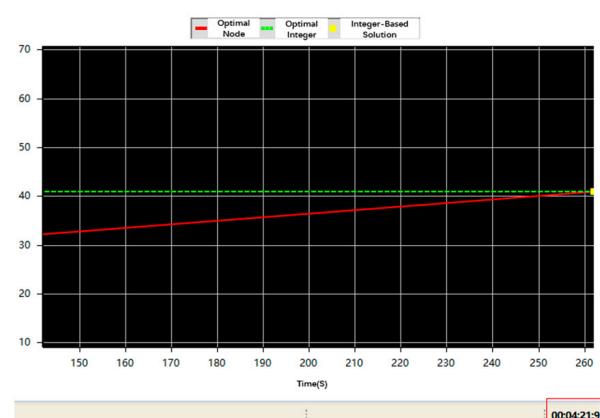


Figure 1. Convergence plot for the CPLEX solution of the n5-s2-m2 example.

Arduous proof of optimality: However, after identifying the optimal solution, CPLEX consumes almost all the remaining computational time to slowly elevate the red line representing the ‘best node’ (i.e., the lower bound of the problem). The core task of the solver transitions from ‘searching for a solution’ to ‘proving that the current solution cannot be improved.’ The red line converges to 41 only in the final moments of the computation, thereby mathematically completing the proof of optimality.

This observation strongly demonstrates that for the studied HFSP-SDST-T, the computational bottleneck lies not in finding a high-quality solution but in proving its optimality. Even for an instance with jobs, which is traditionally considered trivial, the computational cost associated with the optimality proof is exceptionally high.

The specific details of the optimal schedule are as follows:

Machine Assignment:

Stage 1: Jobs {1, 2, 4} are assigned to machine  $M_{11}$ ; jobs {3, 5} are assigned to machine  $M_{12}$ .

Stage 2: Jobs {1, 4, 5} are assigned to machine  $M_{21}$ ; jobs {2, 3} are assigned to machine  $M_{22}$ .

Processing Sequence:

On machine  $M_{11}$ , the sequence of operations is  $4 \rightarrow 1 \rightarrow 2$ .

On machine  $M_{12}$ , the sequence of operations is  $3 \rightarrow 5$ . On machine  $M_{21}$ , the sequence of operations is  $4 \rightarrow 1 \rightarrow 5$ .

On machine  $M_{22}$ , the sequence of operations is  $3 \rightarrow 2$ .

To intuitively illustrate the temporal details of this schedule, the corresponding Gantt chart is plotted in Figure 2.

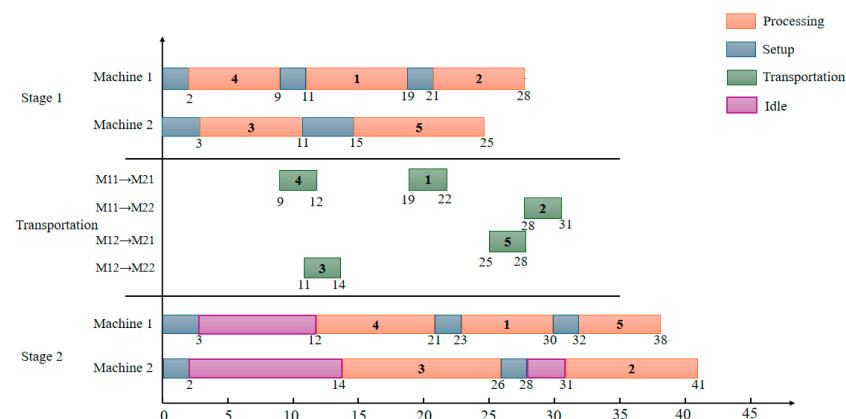


**Figure 2.** Gantt chart of the scheduling plan for the benchmark case in the n5-s2-m2.

It breaks down non-productive time into three distinct components: sequence-related setup time (blue blocks); inter-phase transport time (green blocks); idle time (purple blocks). The exact solution of this instance not only validates the effectiveness of the proposed MILP model but also establishes a crucial optimal benchmark ( $C_{\max} = 41$ ) for evaluating the performance of the subsequent approximate algorithms.

### 2.5. Analysis of Model Characteristics and Computational Bottlenecks

HFSP is inherently NP-hard, and the computational difficulty typically increases exponentially as the problem scale increases. However, the findings in Section 2.4 reveal a deeper and more formidable challenge: the computational bottleneck is not merely scale-dependent but is rooted in the intrinsic and structural complexity of the problem.

This predicament is profoundly reflected in the solving process of benchmark instances. As illustrated in Figure 1, although the optimal solution is rapidly identified in the initial phase, the solver subsequently consumes over 95% of the total computational

time to mathematically prove the optimality. This process, known in operations research as “closing the gap,” involves exhaustive exploration via branch-and-bound to exclude all other potential solutions, constituting the primary cost of the solution process.

The root cause of this phenomenon lies in the strong coupling effects between three features: UPM, SDST, and transportation times. The interleaving of these constraints results in an exceptionally complex solution-space structure. For instance, a sequencing adjustment aimed at optimizing setup times may render the original machine assignment suboptimal, forcing the algorithm to re-evaluate a large number of decision branches. Consequently, traditional exact algorithms struggle to perform effective pruning, leading to a significant decline in search efficiency. Ultimately, even for a small-scale instance with only five jobs, the proof of optimality exhibits typical characteristics of “combinatorial explosion.”

Therefore, it can be concluded that the computational bottleneck for this problem is not the discovery of high-quality solutions but the proof of their optimality. In industrial applications requiring rapid responses, incurring substantial temporal costs to pursue theoretical perfection yields extremely low marginal benefits and is practically unfeasible.

This conclusion provides a powerful motivation for developing novel solution algorithms; an effective paradigm must be designed to circumvent, rather than confront, the bottleneck of optimality proof. The core objective of this paradigm is to rapidly generate high-quality approximate solutions within an acceptable and finite computational time-frame, thereby satisfying the dual requirements of decision efficiency and solution quality in engineering practice. The development of a high-performance metaheuristic is a direct response to this urgent demand and constitutes the core of the subsequent work presented in this study.

### 3. Design of the Adaptive Co-Evolutionary Memetic Algorithm

As revealed in Section 2, the computational bottleneck of the studied problem (HFSP-SDST-T) is deeply rooted in its inherent structural complexity and the strong coupling among its constraints. Consequently, employing exact algorithms to solve medium- to large-scale instances is highly impractical. To address this challenge, this study shifts the paradigm from exact methods to intelligent optimization by proposing a novel ACMA.

#### 3.1. Overall Framework of the Algorithm

The ACMA proposed in this paper is based on a design philosophy that integrates co-evolution, role allocation, and responsive enhancement. By constructing a two-population search system with complementary functions, it achieves a dynamic balance between global exploration, local development, and overcoming local optima.

ACMA consists of two functionally differentiated sub-populations: the first is a global exploration population based on GA, primarily responsible for maintaining diversity in the solution space and continuously generating new solutions with structural differences; the second is an intensive development population based on PSO, primarily responsible for rapid convergence and in-depth exploration around high-quality regions. To prevent functional degradation or search overlap between the two subpopulations during long-term evolution, this paper further designs a periodic coordination mechanism. Through elite information exchange and dynamic role reassignment, the search responsibilities of the two subpopulations are continuously adjusted and strengthened.

Building on this foundation, to address the stagnation phenomenon commonly found in strongly coupled scheduling problems, ACMA incorporates a responsive meme-enhancement mechanism at the top level. When the system detects that the search has entered a state of effective stagnation, the algorithm sequentially triggers localized intensive search and population restructuring operations to achieve a progressive intervention that

transitions from localized optimization to global breakthrough. Consequently, ACMA forms a unified solution framework that integrates global exploration, localized development, collaborative restructuring, and a response to stagnation. Figure 3 shows the overall technical framework of the algorithm.

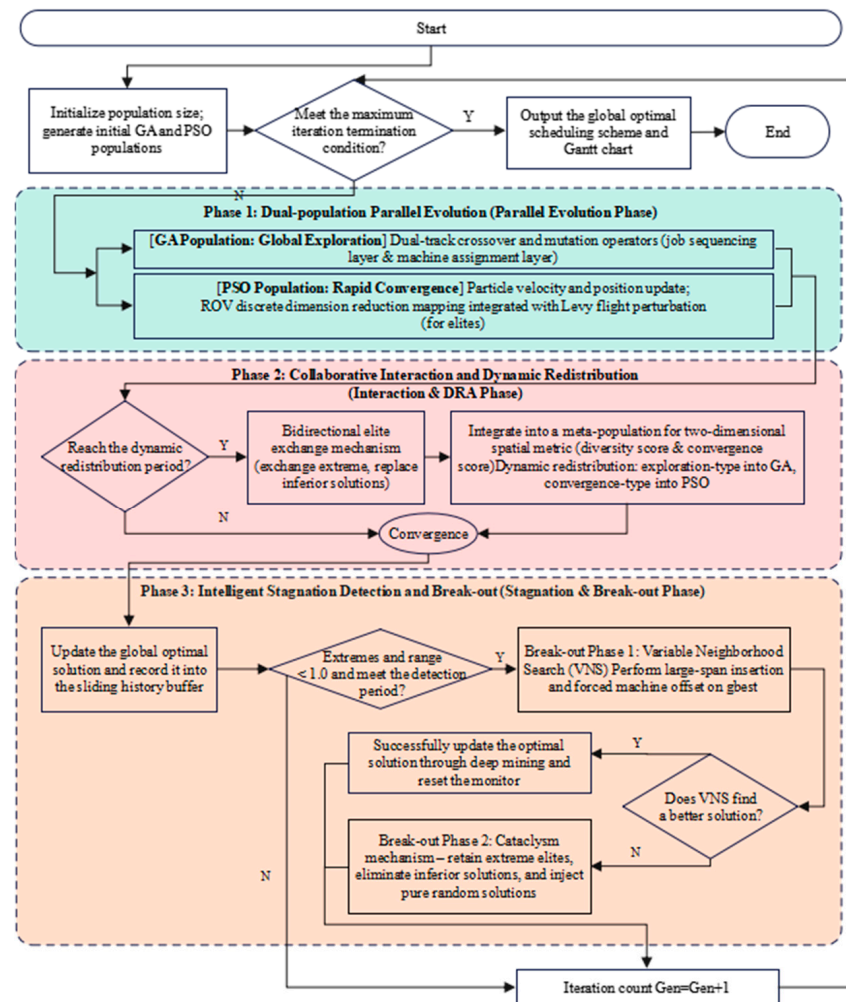


Figure 3. Overall technical framework of ACMA.

The primary innovation of the proposed ACMA lies in an adaptive optimization framework tailored to the multi-constraint structure of the HFSP-SDST-T, which comprises two core mechanisms: (1) A dynamic role assignment mechanism. Unlike traditional static-population parallel evolutionary algorithms, this mechanism evaluates individual diversity and convergence to calculate an adaptability score for each individual based on these two metrics. In descending order of adaptability scores, individuals with higher scores are reassigned to the new-generation global exploration subpopulation, while those with lower scores are reassigned to the new-generation local exploitation subpopulation, thereby dynamically reassigning individual roles. (2) A two-stage memetic enhancement mechanism, which serves as a responsive progressive enhancement mechanism. It is activated when the population is detected to enter an effective state of stagnation. The first-stage variable neighborhood local intensification search is executed first; if this search fails to yield a superior solution, the second-stage population reconstruction is further implemented. Stagnation detection is reset upon completion of the two stages.

Furthermore, several existing methodologies are adapted in the proposed algorithm. The dual-population co-evolution is improved by introducing a cross-population informa-

tion sharing mechanism, as shown in Equation (17): the variable  $x^*$  is no longer restricted to the PSO population, but adopts the historical best position across the dual populations. In terms of adaptive parameter control, the nonlinear time-varying strategies for crossover and mutation probabilities are developed by drawing on and refining classical parameter control theory to further maintain the dynamic balance of the evolutionary process.

### 3.2. Solution Representation and Decoding Process

To simultaneously represent the job sequencing and machine assignment decisions in the HFSP-SDST-T, a two-layer combinatorial encoding scheme is adopted to represent the candidate solutions. Specifically, each individual consists of two components: the first describes the global processing sequence of jobs (jobseq), and the second specifies the machine assignment scheme for each job at each stage (machineassign). A feasible solution can be represented as {jobseq, machineassign}, where jobseq is a  $1 \times n$  set and machineassign is a  $j \times n$  matrix. The former determines the priority order in which jobs enter the schedule construction procedure during decoding, whereas the latter determines the specific parallel machine selected for each operation at the corresponding stage of the job. This representation naturally captures the joint “sequencing-assignment” decision structure in the UPM environment and facilitates the subsequent design of targeted evolutionary operators.

The objective value of any candidate solution is evaluated using a schedule-simulation-based decoding mechanism. During decoding, operations are scheduled stage by stage according to the given job sequence, while the following constraints are jointly considered at each stage: completion and transportation arrival times from the preceding stage, availability time of the assigned machine, and SDST determined by the combination of adjacent jobs on that machine. For a given operation, its actual starting time is determined by both the job arrival time and the time at which the machine becomes available after setup; that is, the maximum of these two values is taken, and its completion time is then determined by the actual starting time and the processing time on the corresponding machine. By recursively proceeding through all stages, the completion times of all jobs at the final stage can be obtained, and the latest completion time is defined as the makespan of the candidate solution,  $C_{\max}$ . This decoding strategy directly embeds the key constraints associated with the UPM, SDST, and transportation times, thereby avoiding complex explicit feasibility-repair procedures and enabling the evolutionary algorithm to reliably evaluate the quality of structurally distinct solutions within a unified encoding space.

### 3.3. Customized Global Explorer

As the global explorer of the algorithm, the core responsibility of the GA is to continuously maintain population diversity across the broad search space. It must also generate new solutions with potential for improvement through structural recombination. The two decision components in the studied problem, job sequencing and machine assignment, are relatively independent in structure. However, they are jointly coupled to the final scheduling outcome through setup times and transportation times. Customized designs are developed for selection, crossover, mutation, and elitist preservation, based on the two-layer encoding structure.

**Selection:** To maintain exploration capability while applying appropriate evolutionary pressure, tournament selection is used. The tournament size is fixed at 3. Three candidate individuals are randomly sampled from the population each time, and the individual with the best fitness is selected as a parent. Using a tournament size of 3 provides slightly stronger, yet still moderate, selection pressure compared with the more common binary tournament. This approach can effectively eliminate poor-quality individuals without

significantly harming diversity. It supplies more competitive genetic material for crossover and mutation operations.

**Dual-track evolutionary operators:** Recognizing the differences between the job sequencing decision and machine assignment decision in terms of encoding structure and optimization role, dedicated genetic operators are designed for these two components. For job sequencing encoding (jobseq), order crossover and adjacent-swap mutation are employed. Order crossover preserves valuable relative order information from parents while generating new sequence recombinations through positional restructuring. Adjacent-swap mutation is implemented by randomly selecting two neighboring positions,  $k$  and  $k + 1$ , and exchanging jobs at these positions. For machine assignment encoding (machineassign), uniform crossover and constrained random-reset mutation are adopted. Uniform crossover recombines different assignment schemes by independently selecting the machine assignment value at each position from one of the two parent chromosomes. In the mutation phase, a constrained random reset is applied to selected positions, where the corresponding gene is reassigned to a legal machine index chosen from the set of available machines at that stage. Through this dual-track design, the GA population can continuously generate new solutions from two complementary perspectives of sequencing-structure recombination and machine-assignment adjustment, thereby enhancing its broad search capability in the complex coupled solution space.

In addition, the crossover probability  $p_c$  and mutation probability  $p_m$  are not fixed, but are dynamically adjusted according to the current algebra. Referring to the adaptive parameter control idea of Srinivas et al. [35], the quadratic dynamic attenuation strategy is adopted for the crossover probability  $p_c$ , as shown in Equation (15).

$$P_c(g) = P_c^{\max} - (P_c^{\max} - P_c^{\min})\left(\frac{g}{G_{\max}}\right)^2 \quad (15)$$

where  $g$  denotes the current generation,  $G_{\max}$  denotes the maximum number of generations, and  $P_c^{\max}$  and  $P_c^{\min}$  represent the upper and lower bounds of the crossover probability, respectively. The purpose of this strategy is to shape the search rhythm of the GA itself: a high crossover probability is maintained in the early stage of evolution to promote large-scale genetic recombination and diversified exploration; as evolution proceeds, the crossover probability gradually decreases to strengthen the preservation of high-quality structural patterns already formed, thereby naturally shifting the search focus toward more refined structural optimization.

Based on the non-monotonic dynamic parameter control concept proposed by Karaftas and Hoogendoorn [36], a periodic pulsed non-monotonic mutation probability following a sine-square function is designed for the mutation probability  $p_m$ , as detailed in Equation (16).

$$P_m(g) = P_m^{\min} + (P_m^{\max} - P_m^{\min}) \sin^2\left(\pi \frac{g}{T_m}\right) \quad (16)$$

where  $P_m^{\max}$  and  $P_m^{\min}$  denote the upper and lower bounds of the mutation probability, respectively, and  $T_m$  is the parameter controlling the pulse period. Unlike monotonically decaying mutation strategies, this mechanism can periodically generate relatively high mutation intensity during the evolutionary process, thereby endowing the GA subpopulation with a certain ability to escape local optima without relying on external intervention.

To ensure that the algorithm retains basic monotonic improvement during global exploration, a mild elitist preservation strategy is introduced. If the best solution in the current generation is inferior to the historical best solution, the historical best solution is forcibly inserted by replacing the worst individual in the current generation.

This strategy preserves elite information while preventing the premature domination of super individuals.

### 3.4. Guided Fast Converger with Lévy Flight Capability

In ACMA, the PSO subpopulation is positioned as a fast converger whose core responsibility is not to continuously expand the search boundary on a global scale, as GA does, but rather to perform highly efficient directed exploitation in the vicinity of already promising regions. The HFSP-SDST-T problem is a typical discrete combinatorial optimization problem, and the standard continuous PSO update mechanism based on real-valued velocity and position superposition cannot be directly applied to the scheduling representation jointly composed of job sequencing and machine assignment. To address this issue, PSO is specifically adapted to the two-layer encoding structure, so that it preserves the fundamental “individual learning–social learning” principle of particle swarm optimization while becoming suitable for intensified local search in a discrete scheduling space.

First, with respect to particle representation and the feasibility of updated solutions, a discrete mapping mechanism consistent with the two-layer encoding is adopted. For the job sequencing component, the relative magnitudes of the continuous particle positions are transformed into the processing order of jobs. For the machine assignment component, the continuous positions are mapped to legal machine indices at each stage through rounding and boundary correction. By means of this treatment, the continuous update results of PSO can be stably converted into feasible solutions, thereby adapting the method to the discrete problem without altering its basic search framework. In other words, PSO still performs “movement” in a continuous space, but its search outcomes ultimately act on the discrete scheduling solution space through a dedicated mapping procedure. Second, to enhance cooperation between the two subpopulations, the social learning term in the PSO is not limited to the best individuals within its own subpopulation. Instead, the historical global optimum solution found by the entire ACMA system during the joint search is used as a source of guidance. Based on the classic particle swarm optimization algorithm particle update formula by Kennedy and Eberhart [37], an improved co-evolution method is applied to the  $x^*$  variable, where it is no longer the historical best position of a single population PSO, but the historical best position in a dual population. The update of each particle’s velocity and position is shown in Equation (17).

$$\begin{aligned} v_i^{g+1} &= \omega v_i^g + c_1 r_1 (p_i - x_i^g) + c_2 r_2 (x^* - x_i^g) \\ x_i^{g+1} &= x_i^g + v_i^{g+1} \end{aligned} \quad (17)$$

where  $p_i$  denotes the historical best position of particle  $i$  and  $x^*$  denotes the historical best position found by the entire co-evolutionary search process of the two subpopulations. As a result, PSO no longer relies solely on internal experience for local convergence; rather, it continuously performs exploitation under the guidance of the best knowledge accumulated by the entire cooperative system. Consequently, potentially high-quality structures discovered by GA in broader regions can be promptly transformed into directional guidance for the subsequent intensified search of PSO, thereby improving the collaborative efficiency of the overall hybrid framework.

To address the tendency of standard PSO toward premature convergence and its difficulty in escaping local optima in complex discrete solution spaces, a Lévy flight-based elite perturbation mechanism is introduced. Specifically, after the regular update in each generation, a Lévy-flight perturbation is applied to the current best particle in the PSO population to explore new breakthrough directions in its vicinity. Unlike ordinary random perturbations, Lévy flights exhibit a heavy-tailed characteristic, featuring predominantly short-step searches with occasional long jumps. Therefore, it maintains the exploitation of

the current promising region while also allowing low-probability long-distance transitions, which help particles escape local attraction basins. If the perturbed solution is better than the current best solution in the PSO subpopulation, it replaces the worst individual in that population; otherwise, the original population structure remains unchanged.

### 3.5. Periodic Cooperative Symbiosis System

To prevent information isolation and search overlap between the GA and PSO subpopulations during long-term parallel evolution, a periodic cooperative symbiosis mechanism is designed. Triggered at a predefined cooperative interval  $T$ , this mechanism maintains knowledge flow and the vitality of persistent search between the two subpopulations via bidirectional migration at the information level and DRA at the physical level.

First, the algorithm performs bidirectional elite migration between the GA and PSO subpopulations, whereby high-quality individuals generated during their respective evolutionary processes are injected into the other subpopulation. For GA, this operation helps absorb high-quality convergent structures from PSO, thereby improving the starting point for subsequent recombination operations. For PSO, the migrated individuals from GA provide new structural directions, preventing exploitation from remaining confined to a narrow region for an extended period. Through this bidirectional migration, a stable information channel is established between the two subpopulations, enabling mutual utilization.

After completing the individual migration, the algorithm proceeds to the DRA phase. Unlike traditional dual-population algorithms, DRA reassigns their roles based on a joint evaluation of individual convergence and structural diversity. The complete execution logic is presented in Algorithm 1. Its core concept lies in temporarily merging the two subpopulations into a meta-population  $P$  of size  $N$  and dynamically reshaping each individual's subsequent role by jointly evaluating their convergence potential and structural novelty.

---

#### Algorithm 1 Dynamic Role Assignment (DRA) Mechanism

---

**Input:** Combined population  $P = P_{GA} \cup P_{PSO}$  with size  $N$ ; Weights  $\omega_c, \omega_d, \omega_{seq}, \omega_{mach}$

**Output:** Reallocated populations  $P_{GA}$  and  $P_{PSO}$

- 1 For  $u = 1$  to  $N$  do
  - 2 Evaluate convergence (makespan):  $F_u \leftarrow g(X_u)$
  - 3 Calculate dual-dimensional diversity  $D_u$  against all  $X_v \in P \setminus \{X_u\}$
  - 4 
$$D_u \leftarrow \frac{1}{N-1} \sum_{\substack{v=1 \\ v \neq u}}^N (\omega_{seq} d_{seq}(X_u, X_v) + \omega_{mach} d_{mach}(X_u, X_v))$$
  - 5 End for
  - 6 For  $u = 1$  to  $N$  do
  - 7 Normalize convergence:  $F_u^{norm} \leftarrow (F_u - F_{min}) / (F_{max} - F_{min})$
  - 8 Normalize diversity:  $D_u^{norm} \leftarrow (D_u - D_{min}) / (D_{max} - D_{min})$
  - 9 Compute suitability score:  $Score_u \leftarrow \omega_d \cdot D_u^{norm} + \omega_c \cdot F_u^{norm}$
  - 10 End for
  - 11 Sort population  $P$  in descending order according to  $Score_u$
  - 12 Update GA population (exploration):  $P_{GA} \leftarrow \{X_u \in P \mid u \leq N/2\}$
  - 13 Update PSO population (exploitation):  $P_{PSO} \leftarrow \{X_u \in P \mid u > N/2\}$
- 

The convergence of individual  $X_u$  is directly characterized by its objective function value. Since this problem aims to minimize the objective, a smaller value implies that the

individual is closer to the current extremum region in the objective space, thus possessing a higher exploitation value (lines 2 and 7 of Algorithm 1).

Given the two-layer encoding scheme, a two-dimensional structure is similarly adopted to evaluate diversity, specifically defining the job sequencing distance and the machine assignment distance. For any two arbitrary individuals  $X_u$  and  $X_v$ :

$$d_{seq}(X_u, X_v) = \frac{\sum_{i=1}^n |\Gamma(X_u, i) - \Gamma(X_v, i)|}{n^2/2} \quad (18)$$

where  $n$  is the total number of jobs, the respective variable  $\Gamma(X_u, i)$  denotes the absolute position of job  $i$  in the sequencing of individual  $X_u$ , the respective variable  $\Gamma(X_v, i)$  denotes the absolute position of job  $i$  in the sequencing of individual  $X_v$ , and the denominator  $n^2/2$  represents the theoretical maximum positional deviation.

$$d_{mach}(X_u, X_v) = \frac{1}{j \cdot n} \sum_{i=1}^n \sum_{s=1}^j \Xi[A(X_u, i, s) \neq A(X_v, i, s)] \quad (19)$$

where  $j$  is the total number of stages,  $s$  is the stage index, and the corresponding variable  $A(X_u, i, s)$  denotes the machine code actually assigned to job  $i$  at stage  $s$  in the decoding scheme of the individual  $X_u$ ;  $A(X_v, i, s)$  denotes the machine code actually assigned to job  $i$  at stage  $s$  in the decoding scheme of the individual  $X_v$ . An indicator function  $\Xi[\cdot]$  is employed, which takes the value of 1 if the condition is satisfied and 0 otherwise. The denominator represents the total number of machine assignment decisions contained within a complete scheduling solution.

After normalizing the two metrics, line 9 of Algorithm 1 combines the exploration and exploitation tendencies into a single score using a weighted sum. Here,  $\omega_d = 0.6$  and  $\omega_c = 0.4$ . This fitness scoring function is designed with a clear focus on resource allocation: individuals with higher scores typically exhibit characteristics of “extremely high diversity or poor convergence.” By sorting the individuals in descending order of score and applying a physical cutoff (lines 11–13 of Algorithm 1), the system forces those distributed in under-explored regions into the GA subpopulation for thorough global exploration; conversely, individuals with lower scores (structurally similar and of high quality) are assigned to the PSO subpopulation, where their highly targeted guidance mechanism is utilized to perform precise local convergence. This dynamic adaptive matching process ensures that the algorithm consistently maintains a dynamic balance between broad-scale exploration and deep mining. It should be emphasized that the DRA mechanism is a heuristic population management strategy, rather than a theoretically optimal allocation rule. Its design is based on the dynamic balance between “global exploration” and “local development” in the evolutionary process, rather than mathematical guarantees.

### 3.6. Two-Stage Memetic Enhancement Mechanism

In the HFSP-SDST-T, UPM, SDST, and transportation times induce strong nonlinear coupling effects within the objective function expressed in Equation (2). This coupling frequently leads to a sharp decline in population diversity and evolutionary stagnation during the middle and later stages of the search. To address this bottleneck, a state-aware “two-stage responsive progressive enhancement mechanism” is designed to break the optimization deadlock at a minimal computational cost. Its complete logic is outlined in Algorithm 2.

**Algorithm 2** Intelligent Stagnation Detection and Two-Phase Breakthrough Mechanism

**Input:** Population  $P$  with size  $N$ , Global best solution  $X^*$ , Stagnation buffer  $Buf$ , Window size  $W$ , Disturbance ratio  $\beta$ , Max VNS iterations  $K$ , Threshold  $\varepsilon$

**Output:** Updated  $X^*$  and Population  $P$

```

1   $Buf \leftarrow Buf \cup \{g(X^*)\}$ 
2  If current generation  $gen \pmod W = 0$  and  $(\max(Buf) - \min(Buf)) < \varepsilon$  then
3    /Phase 1: Variable Neighborhood Search (Rescue)/
4     $X_{temp} \leftarrow X^*$ 
5    For  $r = 1$  to  $K$  do
6      Generate neighbor  $X_{new}$  via Job-Insertion operator  $N_{ins}(X_{temp})$ 
7      If  $g(X_{new}) < g(X_{temp})$  then  $X_{temp} \leftarrow X_{new}$ ; continue;
8      Generate neighbor  $X_{new}$  via Machine-Change operator  $N_{mach}(X_{temp})$ 
9      If  $g(X_{new}) < g(X_{temp})$  then  $X_{temp} \leftarrow X_{new}$ ; continue;
10   EndFor
11   If  $g(X_{temp}) < g(X^*)$  then
12      $X^* \leftarrow X_{temp}$   $Buf \leftarrow g(X^*)$ 
13   Else
14     / Phase 2: Catastrophe Mechanism (Global Rebuilding)/
15     Sort  $P$  in ascending order according to makespan  $g(X)$ 
16      $N_{keep} \leftarrow [N \times (1 - \beta)]$   $P_{elite} \leftarrow \{X_u \in P \mid u \leq N_{keep}\}$ 
17     Generate  $P_{rand}$  containing  $(N - N_{keep})$  fully random individuals
18      $P \leftarrow P_{elite} \cup P_{rand}$ 
19      $Buf \leftarrow g(X^*)$ 
20   EndIf
21 EndIf

```

The core of this mechanism consists of the following three logical tiers:

### 3.6.1. Effective Stagnation Detection

To prevent ineffective interventions during the normal convergence process, the algorithm must possess the capability to accurately identify “stagnation states.” As shown in lines 1–2 of Algorithm 2, the algorithm introduces a sliding observation window  $w$  and a fixed-capacity global optimum buffer matrix  $Buf$ . Specifically, within a fixed-length observation window, the algorithm continuously records the global best values of recent generations, stores them in  $Buf$ , and periodically evaluates the magnitude of variation in the objective values within this window. If the difference between the historical best values recorded in  $Buf$  falls below a minimal tolerance threshold  $\varepsilon$ , it is mathematically determined that the entire population has entered a state of effective stagnation.

### 3.6.2. Phase 1: Local Rescue Based on VNS

Once effective stagnation is detected, the algorithm first activates a low-cost local intensification strategy, attempting to perform a high-intensity fine-grained exploitation on the current global best solution  $X^*$ . As shown in lines 3–10 of Algorithm 2, this phase extracts  $X^*$  and applies VNS logic for targeted structural repair. In light of the dual-layer decision attributes of the scheduling problem, two complementary operators are applied alternately during this phase: the job insertion operator  $N_{ins}$ , which aims to optimize SDST, and the machine reassignment operator  $N_{mach}$ , which seeks to eliminate absolute bottlenecks in machine loads.

Within a maximum of  $K$  trial iterations, the algorithm adopts an intervention strategy characterized by “sequence-first, and restart upon first improvement.” Specifically, the algorithm prioritizes exploration within the  $N$  neighborhood. Once a superior temporary solution  $X_{new}$  is obtained, it is accepted immediately, and the algorithm directly proceeds to the next iteration (line 7 of Algorithm 2). Only when adjustments to the sequencing structure fail to yield improvements does the exploration focus shift to the  $N$  neighborhood in an attempt to rectify the machine assignment scheme (line 8 of Algorithm 2). If the global best solution  $X^*$  is successfully updated during this phase (lines 11–12 of Algorithm 2), it indicates that the current sub-region still possesses potential. Consequently, the enhancement is deemed successful; the system updates the buffer matrix and directly returns to the regular evolutionary process.

### 3.6.3. Phase 2: Population Catastrophic Mutation and Global Rebuilding

If the  $K$  intensive searches in Phase 1 fail to yield any improvement, it indicates that the current  $X^*$  has formed structural convergence at a deeper level; therefore, continuing to invest computational resources into this neighborhood would be futile. In this scenario, the algorithm is forced to trigger Phase 2 (lines 14–20 of Algorithm 2). To strike a balance between ‘retaining high-quality genes’ and ‘forcibly injecting diversity,’ the system sorts the current population  $P$  a priori based on the makespan  $g(X)$ . Based on a preset perturbation ratio  $\beta$ , the algorithm extracts the top  $(1 - \beta)$  proportion of the highest-quality individuals to form an elite set  $P_{elite}$ . The remaining  $\beta$  proportion of individuals is completely eliminated and replaced by an equivalent number of highly diverse random solutions  $P_{rand}$  and resampled from the global feasible solution space. Finally, the algorithm re-enters the co-evolutionary process and conducts a new round of search on the reconstructed population.

### 3.7. Computational Complexity Analysis

To further analyze the computational efficiency of the proposed ACMA, the time complexity of its three major components is discussed separately. Here,  $N$  refers to the population size,  $n$  denotes the number of jobs,  $m$  represents the number of production stages, and  $\beta_2$  stands for the scaling coefficient that caps the maximum iterations of local search. For the DRA, the time complexity of fitness evaluation for all individuals is  $O(N \cdot n \cdot m)$ , the complexity of two-dimensional population diversity calculation is  $O(N^2 \cdot n)$ , and the sorting complexity is  $O(N \cdot \log N)$ . Accordingly, the total complexity of this module is derived as  $T_{DRA} = O(N \cdot n \cdot m + N^2 \cdot n + N \cdot \log N)$ . In the co-evolution phase, GA and PSO updates are executed once per iteration, covering crossover, mutation, particle position update, and fitness evaluation. The corresponding complexity is denoted by  $T_{evo} = O(N(n^2 + n \cdot m))$ . The computational overhead of the two-stage memetic module mainly originates from the VNS local search. Generating a single neighboring solution costs  $O(n^2)$ , while decoding and objective evaluation take  $O(n \cdot m)$ . Given that the maximum number of local search iterations is bounded by  $\beta_2 \cdot n$ , the complexity of this memetic module is  $T_{meme} = O(\beta_2 \cdot n(n^2 + n \cdot m))$ .

Summing up the complexity of all components, the overall time complexity of ACMA is formulated as  $T_{total} = O(N \cdot n \cdot m + N^2 \cdot n + N \cdot \log N + N(n^2 + n \cdot m) + \beta_2 \cdot n(n^2 + n \cdot m))$ .

For large-scale test instances,  $n$  is far larger than both  $N$  and  $m$ . Therefore, the worst-case overall complexity can be simplified to  $O(n^3)$ .

## 4. Computational Experiments and Analysis of Results

This section aims to provide a comprehensive and in-depth evaluation of the proposed ACMA for solving the HFSP-SDST-T through a series of systematic computational experiments. The experimental design follows an inside-out approach. First, by comparing ACMA with an exact algorithm for small-sized instances, the fundamental effectiveness of

ACMA in obtaining high-quality solutions is validated. Second, carefully designed ablation experiments are conducted to systematically examine the necessity and effectiveness of the core internal mechanisms of the algorithm, such as the DRA and two-stage memetic engine. Finally, through a multidimensional comparative analysis against two state-of-the-art algorithms in the field, supplemented by statistical tests, the overall superiority of ACMA in terms of solution quality, convergence efficiency, and robustness is clearly established.

#### 4.1. Experimental Setup

To ensure the fairness, reproducibility, and statistical validity of all experimental results, this section details the test instance set, comparison algorithms, performance metrics, and computational platform.

##### 4.1.1. Test Instance Set Generation

To ensure the generality and credibility of the experimental results, a structured benchmark instance set is constructed in this study. The generation procedure follows the classical Taillard benchmark philosophy for flow shop scheduling and is further extended to incorporate the specific characteristics of the problem under study, namely, UPM, SDST, and inter-stage transportation times. Thus, the generated instances are both challenging and practically relevant.

The instance set covers a wide range of problem scales: the number of jobs is set to  $n \in \{5, 20, 50, 100\}$ , and the number of stages is set to  $s \in \{2, 4, 6\}$ . To simulate the realistic imbalance of production capacities across stages, the number of parallel machines at each stage, denoted by  $M_s$ , is uniformly and randomly selected from the set of integers, that is  $M_s \sim U[2, 4]$ .

The process-related parameters of all instances are generated according to the following stepwise procedure:

**Basic processing times:** The baseline processing time  $p_{i,s}$  follows Taillard's standard distribution, that is,  $p_{i,s} \sim U[1, 99]$ .

**Processing times on unrelated parallel machines:** A machine efficiency factor  $f_{s,k}$  is introduced, and the resulting processing time  $p_{i,s,k}$  is computed by  $p_{i,s,k} = \text{round}(p_{i,s} \times f_{s,k})$ .

**SDST:** To systematically evaluate the algorithm's performance under different setup-time intensities, two levels of setup-time factors are considered, representing scenarios in which setup times account for different proportions of the total time. For the low setup-time intensity case (STF = 0.5), the SDST values are generated from  $U[1, 99 \times 0.5] \approx U[1, 50]$ . For the high setup-time intensity case (STF = 1.25), the SDST values are generated from  $U[1, 99 \times 1.25] \approx U[1, 125]$ . By generating both versions for each problem scale, the algorithm's performance can be examined more comprehensively than before.

**Inter-stage transportation times:** Transportation times are generated within a range comparable to that of processing times, namely  $t_{s,s+1} = U[1, 30]$ .

Using this procedure, for each scale combination, one independent instance is generated for each of the two setup-time intensity levels, resulting in 24 instances. The detailed numerical values for these instances are provided in the dataset.

##### 4.1.2. Comparison Algorithms

To comprehensively and objectively evaluate the overall performance of ACMA and the effectiveness of its internal mechanisms, two levels of comparative experiments were carefully designed in this study, involving four benchmark algorithms.

##### Level 1: Validation of Internal Component Effectiveness

To quantify the contributions of the two core innovative mechanisms in ACMA, two degraded variants of ACMA were constructed for comparison purposes. (1) ACMA-no-

DRA: In this variant, the periodic DRA mechanism is removed. After elite migration between the two subpopulations, no role is reconstructed based on the composite measure of individual diversity and convergence. This experiment was intended to inversely verify the necessity of proactive population restructuring for suppressing premature convergence and maintaining the long-term exploration capability of the system. (2) ACMA-no-ME: In this variant, the responsive two-stage memetic enhancement mechanism is completely removed. This experiment was designed to investigate the critical role of stagnation-detection-driven intensive intervention in enabling the algorithm to escape deep local optima.

## Level 2: Validation of External Competitiveness Against SOTA Algorithms

To establish the leading performance of ACMA in the current scheduling optimization literature, two widely recognized high-performance advanced algorithms were selected as external benchmarks.

**Discrete Artificial Bee Colony (DABC) Algorithm:** As a benchmark algorithm among swarm intelligence methods for discrete scheduling problems, DABC simulates social division of labor and cooperation within a bee colony. Its three-phase search—comprising employed bees, onlooker bees, and scout bees—demonstrates strong global exploration capability in tackling discrete combinatorial optimization problems. This algorithm is selected for comparison to assess ACMA's overall ability in wide-area search and in maintaining population diversity. **Hybrid Ant Colony Optimization (HACO) Algorithm:** This algorithm originates from the scheduling literature [38]. HACO creatively integrates the global positive-feedback pheromone guidance mechanism of Ant Colony Optimization (ACO) with advanced crossover and mutation operators from heuristic algorithms. This hybrid metaheuristic combines a reinforced search mechanism and balances global exploration and local exploitation. Selecting this algorithm for comparison provides a highly convincing test of ACMA's ultimate optimization capability in complex machine environments and under sequence-dependent constraints. To adapt it to the HFSP-SDST-T problem, corresponding modifications are made in this paper. The original no-wait completion-time evaluation component is removed, while its core pheromone-guided sequence generation and genetic operators are retained. The decoding process is reconstructed by building upon the original generation of job sequences, introducing a greedy machine-assignment decoding based on the earliest completion time. Specifically, the completion time of each operation on every available machine is evaluated, with sequence-dependent setup times and transportation time constraints considered simultaneously, ultimately selecting the machine that yields the earliest job completion. In this way, a job sequence is transformed into a corresponding scheduling solution. Notably, all algorithms adopt a unified evaluation function for performance comparison, ensuring fair evaluation under the same criterion and avoiding evaluation bias.

### 4.1.3. Performance Metrics and Computational Environment

The HFSP-SDST-T problem considered in this study involves a wide range of instances, from small-scale to ultra-large-scale. To ensure absolute fairness in the performance evaluation, all competing algorithms were assigned the same stopping criterion, namely, an identical maximum number of iterations. For each test instance, each algorithm was independently executed 10 times. Based on these repeated runs, the following statistical measures were extracted to characterize, in greater detail, the optimization capability and stability of each algorithm within the complex solution space.

**Best:** the best makespan obtained over 10 independent runs. Under the same computational budget, this metric directly reflects the ultimate exploration capability of the algorithm in penetrating the complex solution space and approaching the global optimum.

**Mean:** the arithmetic mean of the makespan values obtained over 10 runs. This metric represents the expected solution quality across repeated runs and serves as the most essential indicator for assessing the overall optimization capability of the algorithm and its ability to escape Local Optima.

**Std:** the standard deviation of the 10 obtained solutions. A smaller Std indicates a stronger resistance to random perturbations. It suggests that the algorithm can not only avoid search stagnation but also converge stably toward high-quality solution regions, thereby demonstrating superior robustness.

All algorithms were implemented and executed on the same computational platform with the following configuration: 13th Gen Intel® Core™ i5-13400F 2.50 GHz, 32 GB RAM, and Windows 10 operating system.

#### 4.2. Parameter Calibration and Validation of Effectiveness

In this section, the key parameters of ACMA are first systematically calibrated using the Taguchi method to ensure that the algorithm operates under a near-optimal configuration, thereby guaranteeing the fairness and validity of the subsequent comparative experiments. Thereafter, by comparing its results with the exact solutions obtained by CPLEX on small-sized instances, the effectiveness of the calibrated algorithm and its ability to obtain high-quality solutions are validated.

##### 4.2.1. Parameter Calibration Based on the Taguchi Method

The performance of a metaheuristic is highly dependent on its parameter settings. To avoid the inefficiency and subjectivity associated with the traditional trial-and-error approach, the Taguchi method, an efficient and robust experimental design technique, is employed to identify the parameter combination that most significantly affects the performance of ACMA.

Based on the in-depth analysis of the core modules presented in Section 3, namely the co-evolutionary framework, the memetic engine, and the Genetic Algorithm, five control parameters that have a decisive influence on the algorithm's exploration, exploitation, cooperation, and stagnation-escape behaviors are identified as the factors in the parameter tuning experiment.

To comprehensively evaluate the effects of these factors and investigate their potential interactions, three discrete levels are assigned to each factor. The selected factors and their corresponding levels are reported in Table 1. These five core factors are described as follows.

**Table 1.** Parameter levels of ACMA.

| Parameter   | Levels |     |     |
|-------------|--------|-----|-----|
|             | 1      | 2   | 3   |
| $P_c^{max}$ | 0.8    | 0.9 | 1   |
| $T$         | 15     | 25  | 35  |
| $W$         | 10     | 20  | 30  |
| $K$         | 300    | 600 | 900 |
| $\beta$     | 0.3    | 0.4 | 0.5 |

$P_c^{max}$ : This factor originates from the adaptive strategy of the GA module. In the proposed design, the crossover probability  $p_c$  varies dynamically throughout the evolutionary process. By optimizing the upper bound  $P_c^{max}$  of its variation range, while fixing the lower bound  $P_m^{min}$  at 0.4, the exploration intensity of the GA during the early search stage can be

effectively controlled. A larger  $P_c^{max}$  facilitates broader search in the initial phase, whereas a smaller  $P_c^{max}$  leads to faster convergence.

$T$ : This factor is derived from the co-evolutionary module and defines the macro-level frequency of information exchange between the GA and PSO populations. The setting of  $T$  directly affects the balance between independent evolution, which preserves the distinctive search characteristics of each subpopulation, and periodic cooperation, which enables the sharing of high-quality information. It is therefore a key parameter governing the overall rhythm of cooperation within the algorithm.

$W$ : This factor serves as the ‘neural center’ of the memetic engine. It determines the memory horizon used by the algorithm to judge whether the search has entered a stagnation state. A smaller  $W$  makes the engine more sensitive and alert to performance stagnation, thereby triggering intervention mechanisms more frequently. In contrast, a larger  $W$  tolerates more short-term fluctuations and thus behaves more conservatively.

$K$ : As the first-stage intervention of the memetic engine, this factor determines the search budget invested by variable neighborhood search in performing local exploitation on the incumbent best solution once stagnation is detected. A larger depth implies stronger local intensification capability and a higher probability of escaping local optima, but it also consumes more computational resources.

$\beta$ : This factor corresponds to the second-stage and more disruptive intervention mechanism of the memetic engine. It controls the proportion of population individuals to be reset when Catastrophic Mutation is triggered after the VNS rescue fails. The  $\beta$  directly determines the amount of newly injected diversity and is a core parameter for breaking deep search deadlocks and achieving creative destruction.

In addition to the above five experimental factors, other non-core parameters, such as the adaptive strategy of the GA mutation probability  $p_m$ , the PSO inertia weight  $\omega$ , the cognitive and social coefficients  $c_1$  and  $c_2$ , the GA tournament selection size, and the number of elites exchanged during cooperation, are fixed at robust conventional values according to extensive practice reported in the literature and preliminary experiments. This ensures that the focus of the calibration remains on the core mechanisms that most strongly influence algorithmic performance.

A full factorial experiment with five factors at three levels each would result in  $3^5 = 243$  parameter combinations. The enormous computational cost makes this approach impractical. Therefore, an L27( $3 \times 13$ ) orthogonal array is employed to efficiently investigate the main effects of each factor. The five experimental factors are mapped to specific columns in the orthogonal array. Experimental configurations are strictly implemented in accordance with the 27-level combinations listed in Table 2. To guarantee result reliability, five independent replicate tests are conducted for each experimental group based on the identical medium-scale test case (n50-s5-m4). Averaged minimum completion times are adopted as the corresponding combination’s response value (Avg). Full experimental design details and obtained results are summarized in Table 2.

**Table 2.** Orthogonal experiments and mean values of ACMA.

| Text | $P_c^{max}$ | $T$ | $W$ | $K$ | $\beta$ | Avg  |
|------|-------------|-----|-----|-----|---------|------|
| 1    | 0.8         | 15  | 10  | 300 | 0.3     | 2234 |
| 2    | 0.8         | 15  | 10  | 300 | 0.4     | 2228 |
| 3    | 0.8         | 15  | 10  | 300 | 0.5     | 2232 |
| 4    | 0.8         | 25  | 20  | 600 | 0.3     | 2223 |
| 5    | 0.8         | 25  | 20  | 600 | 0.4     | 2235 |
| 6    | 0.8         | 25  | 20  | 600 | 0.5     | 2240 |

Table 2. Cont.

| Text | $P_c^{max}$ | $T$ | $W$ | $K$ | $\beta$ | Avg  |
|------|-------------|-----|-----|-----|---------|------|
| 7    | 0.8         | 35  | 30  | 900 | 0.3     | 2242 |
| 8    | 0.8         | 35  | 30  | 900 | 0.4     | 2218 |
| 9    | 0.8         | 35  | 30  | 900 | 0.5     | 2298 |
| 10   | 0.9         | 15  | 20  | 900 | 0.3     | 2177 |
| 11   | 0.9         | 15  | 20  | 900 | 0.4     | 2232 |
| 12   | 0.9         | 15  | 20  | 900 | 0.5     | 2210 |
| 13   | 0.9         | 25  | 30  | 300 | 0.3     | 2247 |
| 14   | 0.9         | 25  | 30  | 300 | 0.4     | 2213 |
| 15   | 0.9         | 25  | 30  | 300 | 0.5     | 2233 |
| 16   | 0.9         | 35  | 10  | 600 | 0.3     | 2194 |
| 17   | 0.9         | 35  | 10  | 600 | 0.4     | 2240 |
| 18   | 0.9         | 35  | 10  | 600 | 0.5     | 2214 |
| 19   | 1.0         | 15  | 30  | 600 | 0.3     | 2203 |
| 20   | 1.0         | 15  | 30  | 600 | 0.4     | 2232 |
| 21   | 1.0         | 15  | 30  | 600 | 0.5     | 2195 |
| 22   | 1.0         | 25  | 10  | 900 | 0.3     | 2196 |
| 23   | 1.0         | 25  | 10  | 900 | 0.4     | 2211 |
| 24   | 1.0         | 25  | 10  | 900 | 0.5     | 2228 |
| 25   | 1.0         | 35  | 20  | 300 | 0.3     | 2170 |
| 26   | 1.0         | 35  | 20  | 300 | 0.4     | 2261 |
| 27   | 1.0         | 35  | 20  | 300 | 0.5     | 2173 |

Experimental results are analyzed via Minitab software Minitab® 22.2.2 (64-bit) is used in this work., with a response table (Table 3) and a main effects plot (Figure 4) generated to quantify and visualize the impact of each factor on algorithm performance.

Table 3. Mean response of ACMA.

| Levels | $P_c^{max}$ | $T$  | $W$  | $K$  | $\beta$ |
|--------|-------------|------|------|------|---------|
| 1      | 2239        | 2216 | 2220 | 2221 | 2210    |
| 2      | 2218        | 2225 | 2213 | 2220 | 2230    |
| 3      | 2208        | 2223 | 2231 | 2224 | 2225    |
| Delta  | 31          | 9    | 18   | 4    | 20      |
| Rank   | 1           | 4    | 3    | 5    | 2       |

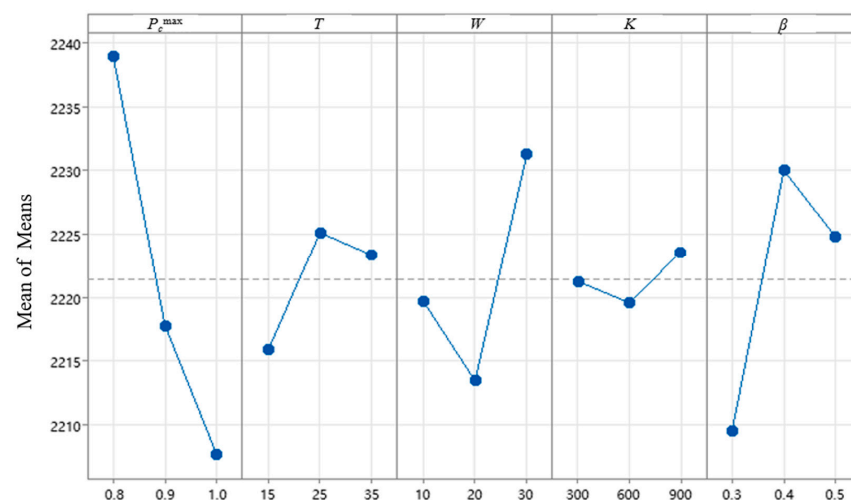


Figure 4. Main effects plot of the mean in ACMA.

Based on the above analytical results, the influence degree of each factor on algorithm performance can be clarified, and the optimal level of each parameter can be determined. Since the minimization of completion time serves as the optimization objective, the lowest point of each factor's curve is selected from the main effects plot in Figure 4. Accordingly, the combination of  $P_c^{\max} = 1.0$ ;  $T = 15$ ;  $W = 20$ ;  $K = 600$ ; and  $\beta = 0.3$ , is confirmed as the optimal parameter configuration.

#### 4.2.2. Validation on Small-Scale Instances

After the core parameters of ACMA are systematically calibrated using the Taguchi method, the next critical step is to verify the algorithm's basic effectiveness. This assessment asked whether ACMA could converge to the desired solution on instances where an optimal solution or high-quality bound is available. For this purpose, a small-scale benchmark, denoted as n5-s2-m2, is chosen. This instance was solved using both CPLEX Studio IDE and ACMA under the best parameter configuration. To eliminate randomness, ACMA is executed independently 10 times. The solution process for CPLEX is shown in Figure 1. A representative convergence profile of ACMA appears in Figure 5. Final performance comparisons between the solvers are reported in Table 4.

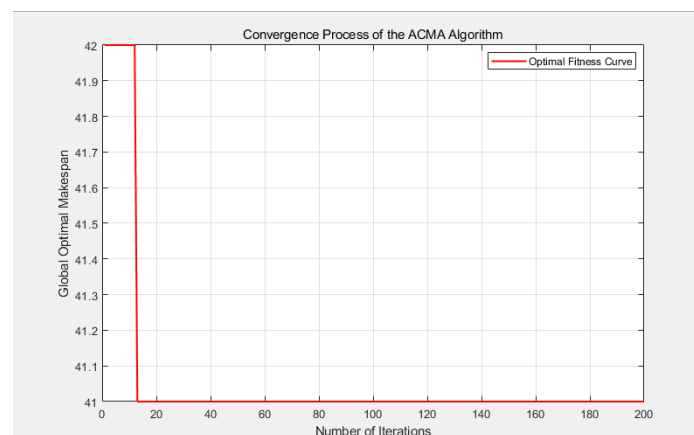


**Figure 5.** Convergence process for ACMA when solving n5-s2-m2.

**Table 4.** Performance comparison of ACMA and CPLEX on the n5-s2-m2 test case.

| Solver | Best Makespan | Average Makespan | Optimal Hit Rate | Average Time |
|--------|---------------|------------------|------------------|--------------|
| CPLEX  | 41            | 41               | 100%             | 261 s        |
| ACMA   | 41            | 41               | 100%             | 2.2 s        |

Several clear observations can be drawn from these results. First, ACMA exhibits solution quality fully consistent with that of CPLEX. Across all 10 independent runs, ACMA achieved a 100% success rate in converging to the optimal solution verified by CPLEX, with a makespan of 41. This provides strong evidence that the proposed adaptive co-evolutionary framework is both correct and highly effective, as it is capable of conducting a precise search in a complex solution space and stably locating the global optimum. Second, the most notable difference between the two solvers lies in computational efficiency. As shown in Figure 5, CPLEX required approximately 261 s to prove the optimal integer solution, whereas ACMA, benefiting from its efficient heuristic search mechanism, obtained the same optimal solution in an average of only 2.2 s, representing an improvement in computational efficiency of approximately 119 times.

At the same time, this result provides compelling empirical support for the computational bottleneck analysis presented in Section 2.5. It clearly demonstrates that, by intelligently bypassing the high-cost optimality-proof process faced by exact algorithms, ACMA is able to preserve optimal solution quality while improving computational efficiency by more than an order of magnitude.

#### 4.3. Core Performance Analysis

##### 4.3.1. Ablation Study: Necessity of the Core Components

To accurately quantify the marginal contributions of the two major innovative mechanisms in ACMA, namely the periodic cooperative symbiosis system and the two-stage memetic enhancement mechanism, a rigorous ablation study is designed and conducted in this section. The complete ACMA is compared with its two degraded variants, ACMA-no-DRA and ACMA-no-ME, which are defined in detail in Section 4.1.2. Representative instances of different scales are selected, with particular emphasis on challenging scenarios characterized by high SDST intensity. Table 5 reports the best solution (Best), mean value (Mean), and standard deviation (Std) obtained by the three algorithms over 10 independent runs on each instance, based on which a comparative analysis is performed.

**Table 5.** Comparative Testing of ACMA and Its Variants.

| Test Instances | ACMA        |               |              | ACMA-no-DRA |        |       | ACMA-no-ME |        |       |
|----------------|-------------|---------------|--------------|-------------|--------|-------|------------|--------|-------|
|                | Best        | Mean          | Std          | Best        | Mean   | Std   | Best       | Mean   | Std   |
| n5-s2-m4       | <b>266</b>  | <b>266</b>    | <b>0</b>     | <b>266</b>  | 268    | 0.94  | <b>266</b> | 267    | 1.41  |
| n5-s4-m4       | <b>560</b>  | <b>561.7</b>  | <b>1.57</b>  | <b>560</b>  | 569.1  | 8.54  | <b>560</b> | 562.9  | 3.03  |
| n5-s6-m4       | <b>753</b>  | <b>753.3</b>  | <b>0.67</b>  | <b>753</b>  | 757    | 5.21  | <b>753</b> | 758.7  | 7.29  |
| n20-s2-m4      | <b>627</b>  | <b>648.1</b>  | <b>12.05</b> | 686         | 710.5  | 22.64 | 631        | 663.7  | 20.42 |
| n20-s4-m4      | <b>1015</b> | <b>1064.4</b> | <b>27.75</b> | 1034        | 1100.6 | 50.92 | 1032       | 1092.3 | 29.19 |
| n20-s6-m4      | <b>1434</b> | <b>1454.2</b> | <b>11.86</b> | 1454        | 1496.7 | 34.78 | 1443       | 1491   | 27.47 |
| n50-s2-m4      | <b>1208</b> | <b>1253.5</b> | <b>24.97</b> | 1261        | 1309.4 | 31.62 | 1282       | 1338   | 28.32 |
| n50-s4-m4      | <b>2565</b> | <b>2625.9</b> | <b>33.92</b> | 2635        | 2752.8 | 55.95 | 2654       | 2732.1 | 45.33 |
| n50-s6-m4      | <b>2891</b> | <b>2949.8</b> | <b>30.45</b> | 2984        | 3089.9 | 68.55 | 3018       | 3062   | 37.99 |
| n100-s2-m4     | <b>2934</b> | <b>3003.2</b> | <b>37.31</b> | 3093        | 3168.7 | 46.41 | 3055       | 3110.8 | 50.06 |
| n100-s4-m4     | <b>4481</b> | <b>4519.2</b> | <b>26.92</b> | 4604        | 4661.1 | 62.65 | 4619       | 4751.1 | 80.55 |
| n100-s6-m4     | <b>5753</b> | <b>5825.7</b> | <b>33.64</b> | 5908        | 6006.1 | 83.85 | 5939       | 5991.9 | 39.75 |

Bold values represent the best-performing result for each test instance in the corresponding row.

As can be seen from Table 5, the complete ACMA consistently achieves the best or near-best overall performance across all test instances, and its advantage becomes increasingly pronounced as the problem scale grows. More specifically, the removal of the memetic enhancement mechanism leads to the most significant performance deterioration. On large-scale instances, both the best and mean values degrade substantially, indicating that the two-stage memetic enhancement mechanism, particularly the deep local exploitation and stagnation-triggered escape strategy, is a critical component for improving solution quality and preventing the algorithm from being trapped in local optima. By contrast, when the DRA mechanism is removed, the algorithm still retains a certain level of solution capability, but its best value, mean value, and variability metrics all deteriorate to varying degrees. Moreover, the increase in standard deviation is more pronounced on large-scale instances, suggesting that the DRA mechanism plays an important role in maintaining population diversity, stabilizing the global exploration process, and enhancing algorithmic robustness. These two mechanisms are therefore complementary and indispensable, jointly underpinning the overall superiority of ACMA.

#### 4.3.2. SOTA Comparison: Comprehensive Performance and In-Depth Analysis Under Specific Constraints

To further validate the performance of ACMA, this subsection compares it comprehensively with DABC and HACO over all 24 benchmark instances. All algorithms are independently executed ten times in the same computational environment. Table 6 reports the best, mean, and standard deviation solutions obtain by each algorithm over the 10 runs.

**Table 6.** Test Results of ACMA and its peers.

| Test Instances  | ACMA |        |       | DABC |        |       | HACO |        |       |
|-----------------|------|--------|-------|------|--------|-------|------|--------|-------|
|                 | Best | Mean   | Std   | Best | Mean   | Std   | Best | Mean   | Std   |
| n5-s2-m4-high   | 266  | 266    | 0     | 266  | 266    | 0     | 266  | 266    | 0     |
| n5-s2-m4-low    | 218  | 218    | 0     | 218  | 218    | 0     | 218  | 218    | 0     |
| n5-s4-m4-high   | 560  | 561.7  | 1.57  | 560  | 561.8  | 1.81  | 562  | 562    | 0     |
| n5-s4-m4-low    | 473  | 474.6  | 1.51  | 473  | 475.2  | 1.23  | 481  | 481    | 0     |
| n5-s6-m4-high   | 753  | 753.3  | 0.67  | 753  | 754.8  | 2.57  | 753  | 753    | 0     |
| n5-s6-m4-low    | 658  | 658    | 0     | 658  | 658.1  | 0.32  | 658  | 658    | 0     |
| n20-s2-m4-high  | 627  | 648.1  | 12.05 | 651  | 659.1  | 6.14  | 639  | 655.5  | 11.18 |
| n20-s2-m4-low   | 520  | 531.6  | 6.26  | 531  | 537    | 3.37  | 533  | 536.4  | 2.17  |
| n20-s4-m4-high  | 1015 | 1064.4 | 27.75 | 1035 | 1074.4 | 20.54 | 1058 | 1078.4 | 13.85 |
| n20-s4-m4-low   | 848  | 874.6  | 18.21 | 856  | 889.4  | 14.25 | 885  | 900    | 8.29  |
| n20-s6-m4-high  | 1434 | 1454.2 | 11.86 | 1472 | 1494.7 | 18.62 | 1441 | 1454.7 | 11.66 |
| n20-s6-m4-low   | 1172 | 1215   | 19.22 | 1216 | 1233.8 | 9.38  | 1194 | 1202.6 | 8.45  |
| n50-s2-m4-high  | 1208 | 1253.5 | 24.97 | 1236 | 1258.7 | 17.94 | 1210 | 1226.7 | 10.12 |
| n50-s2-m4-low   | 982  | 993.5  | 7.53  | 983  | 993.8  | 7.65  | 987  | 1004.4 | 8.80  |
| n50-s4-m4-high  | 2565 | 2625.9 | 33.92 | 2614 | 2643.7 | 19.02 | 2633 | 2755.6 | 55.54 |
| n50-s4-m4-low   | 2029 | 2051.8 | 11.42 | 2031 | 2065.9 | 16.60 | 2169 | 2193.6 | 12.96 |
| n50-s6-m4-high  | 2891 | 2949.8 | 30.46 | 2998 | 3066.5 | 51.63 | 2995 | 3028.8 | 24.58 |
| n50-s6-m4-low   | 2336 | 2371.2 | 36.21 | 2357 | 2383.6 | 17.83 | 2405 | 2437.4 | 19.04 |
| n100-s2-m4-high | 2934 | 3003.2 | 37.31 | 2986 | 3030.3 | 24.05 | 3030 | 3067.7 | 21.15 |
| n100-s2-m4-low  | 2315 | 2347.2 | 15.49 | 2317 | 2356.6 | 18.89 | 2393 | 2429.3 | 16.88 |
| n100-s4-m4-high | 4481 | 4519.8 | 16.49 | 4491 | 4521.8 | 32.12 | 4923 | 5014.2 | 53.63 |
| n100-s4-m4-low  | 3620 | 3645   | 10.36 | 3625 | 3649.3 | 19.48 | 3891 | 3958   | 31.93 |
| n100-s6-m4-high | 5753 | 5825.7 | 33.63 | 5925 | 5976.2 | 35.01 | 6122 | 6197.8 | 34.49 |
| n100-s6-m4-low  | 4646 | 4723.6 | 31.87 | 4677 | 4762.5 | 41.92 | 4959 | 4982   | 14.59 |

Bold values represent the best-performing result for each test instance in the corresponding row.

##### (1) Overall Optimization Capability and Robustness Analysis

The results for the 24 instances clearly show that ACMA exhibits a significant advantage in terms of solution quality over the other two algorithms. On the one hand, the best value typically reflects the ultimate exploration capability of an algorithm under a given computational budget. Across all 24 instances, ACMA directly matches the historically best-known solution in every case, whereas the hit rates of DABC and HACO lag far behind. This demonstrates that ACMA possesses a strong ability to penetrate complex local optima and reach the bottom of the global optimum basin. However, the mean value removes the contingency associated with a single run and is the most essential criterion for evaluating the overall solution capability and average convergence quality of an algorithm. Among the 24 tested instances, ACMA achieves the best mean performance on 21 instances, whereas DABC and HACO do so on only 2 and 5 instances, respectively.

To intuitively reveal the dynamic distribution characteristics of the algorithms over repeated independent runs, boxplots of the 10-run results were drawn for two representative medium-to-large-scale instances, (n50-s6-m4-high) and (n50-s4-m4-high). As can be clearly observed from Figure 6, a key phenomenon emerges: although the numerical

value of ACMA's standard deviation is higher than that of the competing algorithms on some instances, the entire distribution interval of ACMA is still below, and in some cases completely below, the lower bound of the competitors' best solutions. This means that even the worst solution produced by ACMA remains superior to the better solutions obtained by the competing algorithms. A closer examination of the data in Table 6 further illustrates this. For the instance (n5-s4-m4-low), HACO exhibit a perfect zero standard deviation (Best = 481, Mean = 481), whereas ACMA has a standard deviation of 1.51 (Best = 473, Mean = 474.6). The extremely low standard deviation of HACO is, in fact, symptomatic of deep premature convergence: its search operators lack the ability to escape local optima, causing every run to stagnate in the same suboptimal region. In contrast, the slight fluctuation in ACMA's standard deviation is direct evidence of the effectiveness of its powerful anti-stagnation and broad exploration mechanisms. In other words, ACMA trades a marginal increase in variance for a substantial improvement in mean performance.

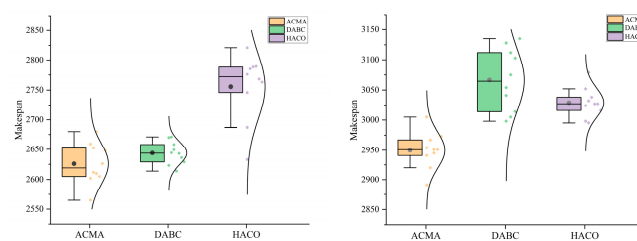


**Figure 6.** Algorithm in a typical example box diagram.

## (2) Statistical Analysis

In order to reliably analyse the significance of differences between algorithms, this section first uses the Anderson–Darling statistical method to test the normality of the experimental data in Table 6, with the results shown in Table 7. The commonly used 5% significance level critical value  $[A^2]_{0.05}$  is 0.759, and the indicators Best, Mean, and Std do not conform to a normal distribution. Therefore, the Wilcoxon signed-rank test is used in this study.

**Table 7.** Anderson–Darling normal distribution test.

| Indicator | [A <sup>2</sup> ] Value |        |        |
|-----------|-------------------------|--------|--------|
|           | ACMA                    | DABC   | HACO   |
| Best      | 2.3024                  | 2.3315 | 2.3482 |
| Mean      | 2.2987                  | 2.3276 | 2.3451 |
| Std       | 1.0543                  | 1.2589 | 1.7654 |

Under the condition of a 95% confidence level ( $\alpha = 0.05$ ), this paper employs the non-parametric Wilcoxon signed-rank test to conduct a paired significance analysis of the performance of ACMA and its comparative algorithms DABC and HACO across three evaluation metrics: Best, Mean, and Std. The results are presented in Table 8.

**Table 8.** Wilcoxon signed-rank test result of different algorithms.

| Indicator | ACMA vs. DABC |         | ACMA vs. HACO |         |
|-----------|---------------|---------|---------------|---------|
|           | Z Value       | p Value | Z Value       | p Value |
| Best      | −3.724        | <0.001  | −3.920        | <0.001  |
| Mean      | −4.107        | <0.001  | −3.302        | <0.001  |
| Std       | −0.146        | 0.884   | −1.164        | 0.224   |

As can be seen from Table 8, pairwise tests of ACMA against the DABC and HACO algorithms on the Best metric yield Z-values of  $-3.724$  and  $-3.920$ , respectively, with both  $p$ -values less than  $0.001$ . A  $p$ -value below  $0.05$  indicates a statistically significant difference. Accordingly, ACMA significantly outperforms DABC and HACO in terms of the Best metric, featuring a better lower bound of optimization and a stronger capability to explore high-quality solutions. For the Mean metric, pairwise comparisons between ACMA and the two benchmark algorithms produce Z-values of  $-4.107$  (vs. DABC) and  $-3.302$  (vs. HACO), where all  $p$ -values are also smaller than  $0.001$ . This demonstrates that ACMA achieves statistically significant superiority in overall average solution quality, delivering more favorable comprehensive optimization performance. In the comparison of the Std metric, the Z-values for ACMA versus DABC and HACO are  $-0.146$  and  $-1.164$ , with corresponding  $p$ -values of  $0.884$  and  $0.244$ . Since both  $p$ -values exceed  $0.05$ , there is no statistically significant difference between ACMA and the two comparative algorithms regarding the fluctuation degree of their optimization results. In summary, ACMA achieves significant improvements in optimization capacity and average solution quality. Such performance gains do not come at the cost of algorithm stability; ACMA maintains robustness comparable to its counterparts and can consistently generate high-quality solutions with low result fluctuations.

### (3) Theoretical Analysis

The HFSP-SDST-T is characterized by strong coupling among unrelated parallel machine assignment, job sequencing, sequence-dependent setup times, and transportation times. These interactions result in a large search space, numerous local optima, and dynamically changing requirements for exploration and exploitation throughout the optimization process. Therefore, an effective algorithm should adapt its search strategy across different evolutionary stages to maintain an appropriate balance between global exploration and local exploitation.

To address these characteristics, the DRA mechanism adjusts individuals' search roles based on their evolutionary states. Individuals with stronger exploration potential remain in the global exploration subpopulation, while those with better convergence characteristics are assigned to the local exploitation subpopulation. This adaptive role transition improves the utilization of search resources, maintains the balance between exploration and exploitation, and reduces the risk of premature convergence.

However, dynamic role assignment alone cannot completely eliminate evolutionary stagnation in such a complex search landscape. Therefore, a responsive progressive enhancement strategy is further introduced to adaptively trigger different search interventions according to the population state. When mild stagnation is detected, VNS is first applied for local intensification. If no further improvement is achieved, a catastrophe strategy is subsequently activated to restore population diversity. Compared with conventional periodic local search, this state-aware strategy adaptively adjusts the enhancement intensity according to the optimization state, reducing unnecessary computational cost while improving the ability to escape deep local optima.

Overall, the DRA mechanism dynamically balances exploration and exploitation, whereas the responsive progressive enhancement strategy provides adaptive intervention against evolutionary stagnation. These two mechanisms complement each other by acting at different stages of the optimization process, enabling ACMA to continuously adjust its search behavior according to the evolutionary state and thereby improve its capability for solving the strongly coupled HFSP-SDST-T.

## 5. Conclusions and Future Work

This study addresses the complex HFSP with UPM, SDST, and inter-stage transportation times, called HFSP-SDST-T, by proposing a novel ACMA. The main contributions are as follows: (1) a precise MILP model with multiple practical constraints was developed to establish an optimal benchmark for heuristic search; (2) a specialized dual-population architecture with an innovative DRA mechanism was proposed, which enabled a shift from passive evolution to proactive population management; (3) a two-stage enhancement engine was designed to improve escape from local optima and ensure stable convergence; (4) extensive test instances confirmed that the algorithm outperforms existing methods in solution quality and robustness.

Extensive computational experiments demonstrated that ACMA effectively handled higher-order coupling and cascading-delay effects induced by multiple physical constraints. In terms of solution quality and cross-scale robustness, it exhibits a degree of superiority over existing state-of-the-art algorithms, thereby providing a highly competitive new tool for intelligent scheduling in modern, complex, flexible shops.

This study is limited to a static single-objective scheduling setting. Future work will be continued by: (1) extending ACMA to a multi-objective framework that considers energy consumption and tardiness; (2) integrating digital twin technology and deep reinforcement learning to address dynamic and uncertain production events.

**Supplementary Materials:** The following supporting information can be downloaded at: <https://www.mdpi.com/article/10.3390/machines14090969/s1>.

**Author Contributions:** Research conception, D.W.; software development, D.W.; formal analysis, D.W. and Y.L.; research implementation, D.W. and Z.Y.; resource support, Y.Z. and Y.W.; Data organization, D.W. and Z.Y.; Writing—first draft, D.W.; Writing—review and editing, W.W.; Supervision, W.W. and G.Y.; Project management, W.W. and G.Y.; Funding acquisition, W.W. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work is supported in part by the National Natural Science Foundation of China (Grant No. 52605825), the Natural Science Foundation of Henan Province (Grant No. 252300423459), the Postdoctoral Fellowship Program of CPSF (Grant No. GZC202323942394), and the Key Research Development and Promotion Project of Henan Province (Grant Nos. 242102220117 and 262102221037), and College Students' Innovative Entrepreneurial Training Plan Program (Grant No. 202610459003).

**Data Availability Statement:** The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

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

## References

1. Yu, B.H. A review of flexible flow shop scheduling problem. *Mod. Manuf. Eng.* **2022**, *504*, 154–162+171. (In Chinese)
2. Tang, H.T.; Yang, Y.; Wen, J. Research on Discrete Manufacturing Workshop Scheduling Problem under Multi-variety and Small-lot Mode. *Digit. Manuf. Sci.* **2022**, *20*, 215–220. (In Chinese)
3. Zhang, C.; Tan, J.; Peng, K.; Gao, L.; Shen, W.; Lian, K. A discrete whale swarm algorithm for hybrid flow-shop scheduling problem with limited buffers. *Robot. Comput.-Integr. Manuf.* **2021**, *68*, 102081. [\[CrossRef\]](#)
4. Han, D.; Tang, Q.; Zhang, Z.; Li, Z. An improved migrating birds optimization algorithm for a hybrid flow shop scheduling within steel plants. *Mathematics* **2020**, *8*, 1661. [\[CrossRef\]](#)
5. Yue, L.; Wang, H.; Mumtaz, J.; Rauf, M.; Li, Z.F. Energy-efficient scheduling of a two-stage flexible printed circuit board flow shop using a hybrid Pareto spider monkey optimization algorithm. *J. Ind. Inf. Integr.* **2023**, *31*, 100412. [\[CrossRef\]](#)
6. Dai, M.; Wang, L.X.; Gu, W.B. Flexible flow shop scheduling problem with batch flow for manufacturing IoT environment. *Trans. Nanjing Univ. Aeronaut. Astronaut.* **2020**, *37*, 831–838.
7. Hasani, A.; Hosseini, S.M.H. A bi-objective flexible flow shop scheduling problem with machine-dependent processing stages: Trade-off between production costs and energy consumption. *Appl. Math. Comput.* **2020**, *386*, 125533. [\[CrossRef\]](#)

8. Zhao, F.; Qin, S.; Yang, G.; Ma, W.; Zhang, C.; Song, H. A factorial based particle swarm optimization with a population adaptation mechanism for the no-wait flow shop scheduling problem with the makespan objective. *Expert Syst. With Appl.* **2019**, *126*, 41–53. [\[CrossRef\]](#)
9. Zhang, B.; Wang, M.Y.; Gao, L. An improved whale swarm algorithm and its application in steelmaking-continuous casting scheduling. *J. Zhengzhou Univ. (Eng. Ed.)* **2018**, *39*, 10.
10. Deng, G.L.; Xu, Z.H.; Gu, X.S. Discrete artificial bee colony algorithm for solving blocked flow shop scheduling. *Chin. J. Chem. Eng.* **2012**, *20*, 1067–1073. (In Chinese) [\[CrossRef\]](#)
11. Xu, T.P.; Zhao, F.Q.; Zhang, J.L. Distributed flexible flow shop scheduling problem with time-sharing tariff constraints and its solution algorithm. *Comput. Integr. Manuf. Syst.* **2023**, 1–20. (In Chinese) [\[CrossRef\]](#)
12. Song, C. A hybrid multi-objective teaching-learning based optimization for scheduling problem of hybrid flow shop with unrelated parallel machine. *IEEE Access* **2021**, *99*, 56822–56835. [\[CrossRef\]](#)
13. Azadeh, A.; Goodrazi, A.H.; Kolaei, M.H.; Jebreili, S. An efficient simulation-neural network-genetic algorithm for flexible flow shops with sequence-dependent setup times, job deterioration and learning effects. *Neural Comput. Appl.* **2019**, *31*, 5327–5341. [\[CrossRef\]](#)
14. An, Y.J.; Kim, Y.D.; Choi, S.W. Minimizing makespan in a two-machine flowshop with a limited waiting time constraint and sequence-dependent setup times. *Comput. Oper. Res.* **2016**, *71*, 127–136. [\[CrossRef\]](#)
15. Khare, A.; Agrawal, S. Scheduling hybrid flowshop with sequence-dependent setup times and due windows to minimize total weighted earliness and tardiness. *Comput. Ind. Eng.* **2019**, *135*, 780–792. [\[CrossRef\]](#)
16. Ozsoydan, F.B.; Sağır, M. Iterated greedy algorithms enhanced by hyper-heuristic based learning for hybrid flexible flowshop scheduling problem with sequence dependent setup times: A case study at a manufacturing plant. *Comput. Oper. Res.* **2020**, *125*, 105044. [\[CrossRef\]](#)
17. Liao, Y.; Li, Z.T.; Li, X.; Peng, C.F. Heuristics for the Hybrid Flow Shop Scheduling Problem with Sequence-Dependent Setup times. *Math. Probl. Eng.* **2022**, *1*, 8682203. [\[CrossRef\]](#)
18. Jemmali, M.; Hidri, L. Hybrid flow shop with setup times scheduling problem. *Comput. Syst. Sci. Eng.* **2023**, *44*, 563–577. [\[CrossRef\]](#)
19. Shen, L.; Dauzère-Pérès, S.; Neufeld, J.S. Solving the flexible job shop scheduling problem with sequence-dependent setup times. *Eur. J. Oper. Res.* **2017**, *265*, 503–516. [\[CrossRef\]](#)
20. Bektur, G.; Saraç, T. A mathematical model and heuristic algorithms for an unrelated parallel machine scheduling problem with sequence-dependent setup times, machine eligibility restrictions and a common server. *Comput. Oper. Res.* **2019**, *103*, 46–63. [\[CrossRef\]](#)
21. Kim, J.G.; Song, S.; Jeong, B.J. Minimising total tardiness for the identical parallel machine scheduling problem with splitting jobs and sequence-dependent setup times. *Int. J. Prod. Res.* **2020**, *58*, 1628–1643. [\[CrossRef\]](#)
22. Liou, C.D.; Hsieh, Y.C. A hybrid algorithm for the multi-stage flow shop group scheduling with sequence-dependent setup and transportation times. *Int. J. Prod. Econ.* **2015**, *170*, 258–267. [\[CrossRef\]](#)
23. Hidri, L.; Elsherbeeney, A.M. Optimal Solution to the two-stage hybrid flow shop scheduling problem with removal and transportation times. *Symmetry* **2022**, *14*, 1424. [\[CrossRef\]](#)
24. Xuan, H.; Liu, S.Y.; Wang, X.Y. Hybrid DABC-GA algorithm for dynamic reentrant flexible flow shop problem with unrelated machines. *Oper. Res. Manag. Sci.* **2022**, *31*, 44–51.
25. Hidri, L.; Tlija, M. Multi-stage hybrid flow shop scheduling problem with lag, unloading, and transportation times. *PeerJ Comput. Sci.* **2024**, *10*, e2168. [\[CrossRef\]](#)
26. Yuan, S.P.; Li, T.K.; Wang, B.L.; Liu, Q. Co-evolutionary iterated greedy algorithm for the two-stage flow shop group scheduling problem. *Syst. Eng.—Theory Pract.* **2020**, *40*, 2707–2716.
27. Mensendiek, A.; Gupta, J.; Herrmann, J. Scheduling identical parallel machines with fixed delivery dates to minimize total tardiness. *Eur. J. Oper. Res.* **2015**, *243*, 514–522. [\[CrossRef\]](#)
28. Nattaf, M.; Artigues, C.; Lopez, P.; Medina, R.; Parada, V.; Pradenas, L. A batch sizing and scheduling problem on parallel machines with different speeds, maintenance operations, setup times and energy costs. In *2015 International Conference on Industrial Engineering and Systems Management (IESM)*; IEEE: Piscataway, NJ, USA, 2016.
29. Cota, L.P.; Guimaraes, F.G.; Oliveira, F.; Souza, M.J.F. An Adaptive Large Neighborhood Search with Learning Automata for the Unrelated Parallel Machine Scheduling Problem. In *2017 IEEE Congress on Evolutionary Computation (CEC)*; IEEE: Piscataway, NJ, USA, 2017.
30. Teng, Y.; Li, X.; Gao, L.; Xie, J.; Li, Y. An Improved Co-Evolutionary Memetic Algorithm Based on Novel Schedule Type and Unconditional Feasibility for Hybrid Flow-Shop Scheduling Problem. *Comput. Ind. Eng.* **2024**, *193*, 110324. [\[CrossRef\]](#)
31. Zhang, W.; Li, C.; Gen, M.; Yang, W.; Zhang, Z.; Zhang, G. A multiobjective memetic algorithm with particle swarm optimization and Q-learning-based local search for energy-efficient distributed heterogeneous hybrid flow-shop scheduling problem. *Expert Syst. With Appl.* **2024**, *237*, 121570. [\[CrossRef\]](#)

32. Fattahi, P.; Saidi, M.M.; Jolai, F. Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *J. Intell. Manuf.* **2007**, *18*, 331–342. [\[CrossRef\]](#)
33. Wang, W.J.; Tian, G.; Luo, M.; Zhang, H.; Yuan, G.; Niu, K. More mixed-integer linear programming models for solving three-stage remanufacturing system scheduling problem. *Comput. Ind. Eng.* **2024**, *194*, 110379. [\[CrossRef\]](#)
34. Wang, W.J.; Yuan, Z.G.; Zheng, W.G.; Cui, K.; Zhang, H.; Guo, X.; Shi, J.; Tian, Z.; Zhang, C.; Tian, G. Integrated optimization framework for human-machine collaborative flexible flow shop scheduling problem based on mixed integer linear programming. *Trans. Beijing Inst. Technol.* **2026**, *46*, 367–378. (In Chinese)
35. Srinivas, M.; Patnaik, L.M. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Trans. Syst. Man Cybern.* **1994**, *24*, 656–667. [\[CrossRef\]](#)
36. Karafotias, G.; Hoogendoorn, M.; Eiben, A.E. Parameter control in evolutionary algorithms: Trends and challenges. *IEEE Trans. Evol. Comput.* **2015**, *19*, 167–187. [\[CrossRef\]](#)
37. Kennedy, J.; Eberhart, R. Particle Swarm Optimization. In *Proceedings of ICNN'95-international Conference on Neural Networks*; IEEE: Piscataway, NJ, USA, 1995; Volume 4, pp. 1942–1948.
38. Engin, O.; Güçlü, A. A new hybrid ant colony optimization algorithm for solving the no-wait flow shop scheduling problems. *Appl. Soft Comput.* **2018**, *72*, 166–176. [\[CrossRef\]](#)

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.