

Article

A Domain-Knowledge-Driven Memetic Algorithm for Energy-Efficient Distributed Flexible Job Shop Scheduling with Machine On–Off Decisions

Li Liu ¹, Chenhao Gu ² and Kaifeng Geng ^{2,*}

¹ School of Digital Media and Art Design, Nanyang Institute of Technology, Nanyang 473000, China; liuli@nyist.edu.cn

² Nanyang Engineering Technology Research Center for Educational Informatization, Center for Informatization Construction and Management, Nanyang Institute of Technology, Nanyang 473000, China; guchenhao@nyist.edu.cn

* Correspondence: gkf@nyist.edu.cn

Abstract

This paper studies a bi-objective distributed flexible job shop scheduling problem considering machine on–off decisions. A mathematical model is formulated to minimize the makespan and total energy consumption while distinguishing processing energy, idle energy, and on–off energy. To address the coupled effects among job-to-factory assignment, machine selection, operation sequencing, and machine on–off states, a domain-knowledge-driven memetic algorithm (DKMA) is proposed. The algorithm represents each schedule with a three-layer encoding scheme and integrates hybrid initialization, knowledge-driven neighborhood search, and energy-saving reconstruction to improve solution-set quality and the use of on–off-eligible idle intervals. The proposed model and algorithm are evaluated through Taguchi parameter tuning, small-scale mixed-integer linear programming (MILP) validation, component ablation experiments, and multi-algorithm comparisons. The results show that DKMA improves solution-set coverage, Pareto-front approximation, and energy control on the tested instances, which supports its applicability to distributed green scheduling with machine on–off decisions.

Keywords: distributed flexible job shop; green scheduling; machine on–off decisions; energy optimization; multi-objective optimization



Academic Editor: Massimiliano Caramia

Received: 11 May 2026

Revised: 24 June 2026

Accepted: 25 June 2026

Published: 30 June 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

Distributed manufacturing has become increasingly common in discrete manufacturing systems under the continuing transition toward green and low-carbon production. Multiple geographically dispersed factories jointly undertake production tasks, which helps alleviate capacity limitations, order fluctuations, and uneven resource allocation in a single factory, and also creates additional opportunities for energy optimization in manufacturing processes. Naderi and Ruiz [1] provided an early systematic study of distributed scheduling and laid the foundation for task assignment and sequencing optimization in multi-factory production environments. Compared with a single-factory shop, a distributed flexible job shop must simultaneously determine job-to-factory assignment, machine selection, and operation sequencing. These decisions may generate scattered or long idle intervals on machine timelines because of imbalanced factory loads, heterogeneous machine choices, and operation waiting. If all such idle intervals are treated as standby periods, unnecessary

non-processing energy consumption will be incurred. Therefore, studying distributed flexible job shop scheduling with machine on–off decisions is important for reducing non-processing energy while maintaining production efficiency, thereby improving the green operation level of multi-factory manufacturing systems.

The flexible job shop scheduling problem (FJSP) allows each operation to be processed on one of several candidate machines; more broadly, job-shop related scheduling problems are known to be NP-hard [2]. Dauzère-Pérès et al. [3] reviewed FJSP studies from the perspectives of problem characteristics, modeling methods, and solution algorithms. On this basis, the distributed flexible job shop scheduling problem (DFJSP) further introduces job-to-factory assignment, extending scheduling from internal optimization within a single shop to collaborative optimization across multiple factories. For DFJSP, Wei et al. [4] incorporated supply-demand matching into scheduling under shared manufacturing; Meng et al. [5] developed mixed-integer linear programming (MILP) and constraint programming (CP) formulations; Luo et al. [6] studied scheduling with inter-factory transfers; Zhang et al. [7] considered crane transportation constraints; and Tang et al. [8] further discussed integrated sequencing flexibility. These studies have extended DFJSP from basic factory assignment, machine selection, and operation sequencing to more complex scheduling scenarios involving transfers, transportation equipment, and sequencing flexibility.

Energy-efficient scheduling is not limited to adding energy consumption as an optimization objective; it also requires identifying the specific energy-saving mechanisms that can be influenced by scheduling decisions. Mouzon et al. [9], Fang et al. [10], Liu et al. [11], and Dai et al. [12] studied this issue from the perspectives of equipment operation control, power consumption and carbon-footprint reduction, total energy minimization in job shops, and energy-efficient scheduling in flexible production systems, respectively. The reviews by Gahm et al. [13] and Fernandes et al. [14] further indicate that scheduling-level energy-saving strategies can generally be grouped into three categories. The first uses time-of-use electricity prices or demand response to place flexible operations in low-price or low-load periods, thereby reducing energy cost [15–17]. The second uses variable machine speed or processing-speed selection to trade off processing time and instantaneous power [18,19]. The third controls machine states during non-processing periods by selecting standby, off, or other low-power states during idle intervals [9,11,18,20–22].

Studies on machine on–off decisions have mainly focused on the coordination between equipment idle management and production scheduling. Mouzon et al. [9] showed that non-bottleneck machines still consume considerable energy in standby mode and reduced equipment operating energy through scheduling rules and on–off policies. Shrouf et al. [20] jointly determined job start times, idle states, and machine on–off times in a single-machine environment with time-of-use electricity prices. Zhang et al. [21] introduced machine on–off control into FJSP and established an energy-efficient scheduling model with total energy consumption and makespan as objectives. Gu et al. [22] further considered transportation and machine on–off constraints in energy-efficient DFJSP. Overall, machine on–off decisions have been shown to be effective for reducing non-processing energy. Although efficient scheduling can reduce idle time, high overall utilization does not mean that every machine is continuously occupied. In distributed flexible job shops, job-to-factory assignment, machine eligibility, technological precedence, and load imbalance may still create medium or long idle intervals on non-bottleneck machines. Therefore, machine on–off decisions are used as a complementary energy-saving mechanism for unavoidable non-processing intervals, rather than as a substitute for workload balancing or utilization improvement. However, existing studies are still concentrated more on single-machine or single-factory job shops, or treat machine on–off control as a specific extended constraint. In contrast, energy-oriented DFJSP studies mainly focus on energy-objective modeling, real-time scheduling,

and metaheuristic or learning-based algorithms [23–27]. The coordinated treatment of on–off decisions with job-to-factory assignment, machine selection, and operation sequencing remains insufficient. In particular, algorithm design still makes limited use of structural information such as critical factories, bottleneck machines, critical operations, and machine idle intervals. Therefore, high-performance algorithms driven by domain knowledge remain worth further investigation for this class of complex scheduling problems.

Table 1 compares representative studies related to DFJSP and energy-efficient scheduling. The comparison shows that existing studies have mainly focused on distributed scheduling decisions, energy-oriented objectives, or machine-state control separately, while the joint consideration of DFJSP, machine on–off decisions, and processing/standby/on–off energy accounting remains limited.

Table 1. Comparison of related scheduling studies.

Ref.	Scheduling Problem	Objectives	Energy-Saving Strategy Considered	Solution Approach
[4]	DFJSP with supply-demand matching	Total cost and makespan	Not energy-oriented	Hybrid estimation-of-distribution and tabu-search algorithm
[5]	DFJSP	Makespan	Not energy-oriented (makespan-oriented)	MILP and constraint programming formulations
[6]	DFJSP with inter-factory operation transfers	Makespan, maximum workload of factories, and total energy consumption	Processing energy consumption, energy consumption of transfer between machines and factories	Efficient memetic algorithm
[7]	DFJSP with crane transportation	Makespan and total energy consumption	Processing energy consumption and energy consumption for three stages of crane operation (i.e., accelerated start-up, uniform motion, and decelerated braking)	Q-learning-based hyper-heuristic evolutionary algorithm
[8]	DFJSP with integrated sequencing flexibility	Makespan and total energy consumption	Energy consumption of processing and processing intervals (machine idling at low speed and machine power-off/power-on operation), and additional energy consumption	Effective memetic algorithm
[9]	Manufacturing-equipment scheduling with underutilized non-bottleneck machines	Energy consumption	Turn-off decisions for underutilized machines during long idle periods	Dispatching rules and a multi-objective mathematical programming model
[10]	Flow-shop scheduling with operation-speed decisions	Makespan, the peak total power consumption, and the carbon footprint	Operation-speed selection for power, energy, and carbon reduction	Mathematical programming model
[11]	Classical job shop scheduling	Total electricity consumption and total weighted tardiness	Basic energy consumption, runtime energy consumption, and cutting energy consumption	NSGA-II

Table 1. Cont.

Ref.	Scheduling Problem	Objectives	Energy-Saving Strategy Considered	Solution Approach
[12]	Flexible flow-shop scheduling	Makespan and total energy consumption	Turn-off/on energy consumption and run-production-mode energy consumption	Improved genetic-simulated annealing algorithm
[15]	FJSP under time-of-use electricity pricing	Total energy cost	Time-of-use pricing scheme	Iterative tabu search algorithm
[16]	FJSP under time-of-use electricity pricing and scheduled downtime	Makespan and total energy cost	Time-of-use pricing scheme and scheduled-downtime-aware scheduling	Integer linear programming and constraint programming models
[17]	FJSP	Makespan, total energy consumption, total energy cost, and peak demand	Time-of-use pricing scheme	Graph reinforcement learning-based method
[18]	FJSP	Makespan, energy consumption, and the number of turning-on/off machines	Machine turn-on/off timing and processing speed-level selection	NSGA-II with a green scheduling heuristic
[19]	Job shop scheduling	Total weighted tardiness and total energy consumption	Processing speed-level selection	Multi-objective genetic algorithm
[20]	Single-machine scheduling under variable energy prices	Total energy consumption cost	Launch-time, idle/shutdown, and turn-on/off decisions under variable prices	Genetic algorithm
[21]	FJSP	Total energy consumption and makespan	Switch-off/switch-on mechanism for long idle intervals	NSGA-II-based solution approach
[22]	DFJSP with transportation and start-stop constraints	Makespan and energy consumption	Transportation-time and machine start-stop coordinated scheduling	Improved memetic algorithm
[23]	DFJSP	Makespan and energy consumption	Full-active scheduling decoding to reduce energy consumption	Surprisingly popular-based adaptive memetic algorithm
[24]	DFJSP	Makespan and total energy consumption	Processing and idle energy consumption	Inverse model and adaptive neighborhood search-based cooperative optimizer
[25]	Dynamic DFJSP with finite AGV transportation resources	Makespan and total carbon emissions	Total carbon emissions are composed of machine work, machine idle, AGV load transportation, and AGV no-load transportation	Deep reinforcement learning-based memetic algorithm
[26]	DFJSP	Energy consumption	Turn-off/on strategies for idle-period energy reduction	Hybrid shuffled frog-leaping algorithm
[27]	DFJSP considering real-time scheduling	Hierarchical multi-objective optimization, including energy consumption	Cutting, idle, tool-changing, and workpiece setup energy consumption	Evolutionary game-based solver method

This paper, therefore, studies a bi-objective distributed flexible job shop scheduling problem with machine on–off decisions (DFJSP-OO). The model minimizes the makespan and total energy consumption, and the solution method is built around a domain-knowledge-driven memetic algorithm. The main contributions are as follows.

(1) A bi-objective mathematical model is developed for distributed flexible job shop scheduling with machine on–off decisions, in which processing energy, idle energy, and on–off energy are modeled simultaneously.

(2) A domain-knowledge-driven memetic algorithm is proposed. It combines three-layer encoding, hybrid initialization, structured neighborhood search, and energy-saving reconstruction to jointly optimize job-to-factory assignment, machine selection, operation sequencing, and on–off-eligible idle intervals.

(3) The effectiveness and stability of the proposed model and algorithm are evaluated through Taguchi parameter tuning, exact MILP validation on a small-scale instance, component ablation experiments, statistical tests, and representative case analysis.

The remainder of this paper is organized as follows. Section 2 describes the problem and formulates the mathematical model. Section 3 presents the DKMA. Section 4 reports the numerical experiments and result analysis. Section 5 concludes the paper and discusses future research directions.

2. Problem Description and Mathematical Modeling

2.1. Problem Description

In DFJSP-OO, jobs are processed in homogeneous distributed factories, and each factory is equipped with m machines indexed by k . Each job i consists of several ordered operations, and different jobs may contain different numbers of operations. To avoid ambiguity, the operation set of the job i is denoted by Ω_i , and the complete operation set is denoted by Ω . The set Ω is the union of all job-specific operation sets, and each globally unique operation $a \in \Omega$ belongs to exactly one job indexed by i_a .

Each operation a can be processed on one machine selected from its eligible machine-index set K_a . If operation a is assigned to the machine k in factory f , its processing time is denoted by p_{afk} . The energy parameters of a machine in the processing, idle standby, and on–off states are denoted by the processing power P^{proc} , idle power P^{idle} , and unit on–off energy e^{sw} , respectively. Here, e^{sw} is defined as the energy consumed by one complete on–off cycle. Since identical energy parameters are used for all machines in this paper, the minimum idle threshold that permits an on–off cycle is uniformly denoted by $T_{off} = e^{sw} / P^{idle}$. The solution must simultaneously determine job-to-factory assignment, operation sequencing, and machine selection, and then determine machine on–off states according to internal idle lengths. Under technological and resource constraints, the makespan and total energy consumption are minimized simultaneously.

The following assumptions are adopted. (1) All factories and machines are available at the start of scheduling, and machines may remain off before their first operation starts. (2) Once an operation starts, it cannot be interrupted. (3) Each job can be assigned to only one factory during the whole processing process, and all of its operations must be completed in that factory. (4) Operations of the same job must be processed sequentially according to the predefined technological route. (5) At any time, each machine can process at most one operation, and each operation can be processed by at most one machine. (6) When the machine idle length is shorter than T_{off} , the machine remains in standby mode; when the idle length reaches or exceeds T_{off} , one complete on–off cycle is triggered. (7) The initial power-on and final power-off are not included in the optimization objective. (8) The energy associated with machine shutdown, startup, warm-up, and setup-related preparation is incorporated into the on–off energy parameter e_{sw} . The processing times used in this study

are treated as effective processing times; when startup, setup, warm-up, or switching time is required, such time is assumed to be included in the given processing time.

The present model focuses on scheduling-level energy components, including processing energy, idle standby energy, and machine on–off energy. Load-dependent power variation, sequence-dependent energy costs, switching-induced wear, and uncertainty in energy parameters are not explicitly modeled in the current deterministic formulation.

2.2. Mathematical Formulation

The symbols used in this paper are defined in Table 2.

Table 2. Symbols used in this paper.

Symbol	Definition
n	number of jobs.
p	number of factories.
m	number of machines in each factory.
i	index of jobs.
f	index of factories.
k	index of machines.
M_k	machine k in a factory.
K	set of machine indices in each factory.
Ω_i	ordered operation set of job i .
Ω	set of all operations, $\Omega = \cup_{i=1}^n \Omega_i$.
a, b	indices of globally unique operations in Ω .
i_a	index of the job to which operation a belongs.
K_a	set of machine indices eligible for processing operation a .
$\mathcal{P}$	set of adjacent operation pairs within the same job.
A_{if}	equals 1 if job i is assigned to factory f ; otherwise, 0.
X_{afk}	equals 1 if operation a is processed on machine k in factory f ; otherwise, 0.
S_a	start time of operation a .
C_a	completion time of operation a .
$C_{\max}$	makespan.
E_{tot}	total energy consumption.
E_{proc}	processing energy consumption.
E_{idle}	idle standby energy consumption.
E_{sw}	on–off-cycle energy consumption.
p_{afk}	processing time of operation a on machine k in factory f .
p^{proc}	processing power of a machine.
p^{idle}	idle standby power of a machine.
e^{sw}	energy consumed by one complete on–off cycle.
Y_{abfk}	sequencing variable for operations a and b on machine k in factory f .
Z_{abfk}	adjacency variable for operations a and b on machine k in factory f .
Z_{afk}^T	equals 1 if operation a is the last processing task on machine k in factory f ; otherwise, it equals 0.

Table 2. Cont.

Symbol	Definition
Z_{afk}^S	equals 1 if operation a is the first processing task on machine k in factory f ; otherwise, it equals 0.
I_{abfk}	idle duration between adjacent operations a and b on machine k in factory f .
U_{abfk}	equals 1 if the idle interval between a and b triggers an on–off cycle; otherwise, 0.
T_{off}	minimum idle threshold for triggering one on–off cycle.
L	a sufficiently large positive number.
ε	a sufficiently small positive number used to characterize the boundary of the on–off threshold.

The optimization objectives and constraints are given as follows.

$$\min C_{\max} \quad (1)$$

$$\min E_{\text{tot}} = E_{\text{proc}} + E_{\text{idle}} + E_{\text{sw}} \quad (2)$$

$$E_{\text{proc}} = \sum_{a \in \Omega} \sum_{f=1, \dots, p} \sum_{k \in K_a} P^{\text{proc}} p_{afk} X_{afk} \quad (3)$$

$$E_{\text{idle}} = \sum_{a \in \Omega} \sum_{b \in \Omega, b \neq a} \sum_{f=1, \dots, p} \sum_{k \in K} P^{\text{idle}} I_{abfk} (1 - U_{abfk}) \quad (4)$$

$$E_{\text{sw}} = \sum_{a \in \Omega} \sum_{b \in \Omega, b \neq a} \sum_{f=1, \dots, p} \sum_{k \in K} e^{\text{sw}} U_{abfk} \quad (5)$$

$$\sum_{f=1}^p A_{if} = 1, \quad \forall i = 1, \dots, n. \quad (6)$$

$$\sum_{f=1}^p \sum_{k \in K_a} X_{afk} = 1, \quad \forall a \in \Omega. \quad (7)$$

$$X_{afk} \leq A_{iaf}, \quad \forall a \in \Omega, f = 1, \dots, p, k \in K_a. \quad (8)$$

$$C_a = S_a + \sum_{f=1}^p \sum_{k \in K_a} p_{afk} X_{afk}, \quad \forall a \in \Omega. \quad (9)$$

$$C_a \leq S_b, \quad \forall (a, b) \in P. \quad (10)$$

$$Y_{abfk} + Y_{bafk} \leq X_{afk}, \quad \forall a \neq b, f, k \quad (11)$$

$$Y_{abfk} + Y_{bafk} \leq X_{bfk}, \quad \forall a \neq b, f, k \quad (12)$$

$$Y_{abfk} + Y_{bafk} \geq X_{afk} + X_{bfk} - 1, \quad \forall a \neq b, f, k \quad (13)$$

$$S_b \geq C_a - L(1 - Y_{abfk}) - L(2 - X_{afk} - X_{bfk}), \quad \forall a \neq b, f, k \quad (14)$$

$$S_a \geq C_b - LY_{abfk} - L(2 - X_{afk} - X_{bfk}), \quad \forall a \neq b, f, k \quad (15)$$

$$\sum_{b \in \Omega, b \neq a} Z_{abfk} + Z_{afk}^T = X_{afk}, \quad \forall a \in \Omega, f, k \quad (16)$$

$$\sum_{b \in \Omega, b \neq a} Z_{bafk} + Z_{afk}^S = X_{afk}, \quad \forall a \in \Omega, f, k \quad (17)$$

$$\sum_{a \in \Omega} Z_{afk}^S \leq 1, \quad \forall f, k \quad (18)$$

$$\sum_{a \in \Omega} Z_{afk}^T \leq 1, \quad \forall f, k \quad (19)$$

$$Z_{abfk} \leq Y_{abfk}, \quad \forall a \neq b, f, k \quad (20)$$

$$I_{abfk} \geq S_b - C_a - L(1 - Z_{abfk}), \quad \forall a \neq b, f, k \quad (21)$$

$$I_{abfk} \leq S_b - C_a + L(1 - Z_{abfk}), \quad \forall a \neq b, f, k \quad (22)$$

$$I_{abfk} \leq LZ_{abfk}, \quad \forall a \neq b, f, k \quad (23)$$

$$I_{abfk} \geq T_{off} U_{abfk}, \forall a \neq b, f, k \quad (24)$$

$$I_{abfk} \leq (T_{off} - \varepsilon)(1 - U_{abfk}) + LU_{abfk}, \forall a \neq b, f, k \quad (25)$$

$$U_{abfk} \leq Z_{abfk}, \forall a \neq b, f, k \quad (26)$$

$$C_{\max} \geq C_a, \quad \forall a \in \Omega. \quad (27)$$

Constraint (6) assigns each job to exactly one factory. Constraint (7) assigns each operation to exactly one processing machine selected from its eligible machine-index set. Constraint (8) uses i_a to ensure that each operation is processed only in the factory assigned to its job. Constraint (9) defines the completion time of each operation. Constraint (10) enforces the technological precedence relationship for operation pairs in P . Constraints (11)–(15) determine the processing order of operations assigned to the same machine and prevent processing overlap. Constraints (16)–(19) identify the first operation, last operation, and adjacent processing chain on each machine. Constraints (20)–(23) calculate internal idle intervals only between directly adjacent operations on the same machine. Constraints (24)–(26) determine whether an internal idle interval triggers an on–off cycle according to the threshold T_{off} . Constraint (27) defines the makespan.

3. DKMA Design for Solving DFJSP-OO

The memetic algorithm (MA) is a metaheuristic optimization method that combines population-based evolutionary search with individual local improvement, and it has been widely used for complex combinatorial optimization problems. Compared with traditional evolutionary algorithms that mainly rely on crossover and mutation operators, a memetic algorithm introduces local search during population evolution, which enhances the exploitation of high-quality solutions and improves search efficiency in complex solution spaces.

DFJSP-OO must simultaneously determine job-to-factory assignment, operation sequencing, machine selection, and machine on–off states. The problem, therefore, contains multiple decision layers and a large solution space. Since the makespan is mainly affected by critical factories, bottleneck machines, and critical paths, whereas total energy consumption depends not only on processing arrangements but also on machine idle intervals and their on–off states, generic evolutionary operations alone cannot sufficiently exploit problem-structure information. To improve the use of critical scheduling structures and energy-saving opportunities, this paper proposes a domain-knowledge-driven memetic algorithm. The algorithm represents a schedule through three-layer encoding, improves the initial solution quality through hybrid initialization, and maintains the search range through population evolution. On this basis, knowledge-driven neighborhood search and energy-saving reconstruction strategies are designed for critical factories, bottleneck machines, critical paths, and on–off-eligible idle intervals. These strategies make targeted adjustments to the key components affecting makespan and energy consumption, thereby improving bi-objective optimization performance. The algorithmic procedure is shown in Algorithm 1. The evaluation budget is scaled according to the instance size, since larger instances require more evaluations to avoid insufficient evolution, whereas smaller instances can usually reach a stable search state with fewer evaluations. Therefore, $MaxEval = 100 \times |\Omega| \times p$ is set, where $|\Omega|$ is the number of operations in the instance and p is the number of factories. If the external nondominated archive is not updated for $G_{stall} = 10$ consecutive generations, the newly generated solutions in this period are considered to provide limited additional nondominated information, and continuing the search may lead to low-return evaluations. Therefore, this condition is used to reduce redundant evaluations and computational cost.

Algorithm 1 DKMA for solving DFJSP-OO

Input: Instance I ; number of factories p ; population size ps ; crossover rate pc ; mutation rate pm ; local-search rate pls ; $MaxEval = 100 \times |\Omega| \times p$; stagnation limit $G_{stall} = 10$.

Output: External nondominated archive A .

```

1  Initialize  $eval \leftarrow 0$  and  $stall \leftarrow 0$ 
2  Generate mixed initial population  $P$  by earliest-completion, energy-oriented and
   random initialization
3  Decode and evaluate  $P$ , apply energy-saving reconstruction, set  $eval \leftarrow eval + |P|$ ,
   and let  $P \leftarrow \text{EnvironmentalSelection}(P, ps)$  and  $A \leftarrow \text{UpdateArchive}(\{\}, P)$ 
4  while  $eval < MaxEval$  and  $stall < G_{stall}$  do
5      Assign a nondominated rank and crowding distance to each individual in  $P$ ,
       and initialize the offspring population  $Q \leftarrow \{\}$ .
6      while  $|Q| < ps$  and  $eval < MaxEval$  do
7          select parents  $x_a$  and  $x_b$  from  $P$  by tournament selection
8          if  $rand < pc$  then apply POX to OSL and uniform crossover to FAL and MSL
       end if
9          mutate FAL, OSL and MSL with probability  $pm$ 
10         decode and evaluate offspring, apply energy-saving reconstruction, set  $eval$ 
             $\leftarrow eval + |offspring|$ , and add them to  $Q$ 
11     end while
12     select the first  $\text{ceil}(pls \times ps)$  elite individuals from  $A$ 
13     for each elite individual  $x$  do
14         extract the critical factory, bottleneck machine, latest critical operation,
            critical jobs and on-off-eligible idle intervals from its schedule
15         generate knowledge-driven neighbors by  $N_1$ - $N_8$ , decode and evaluate the
            accepted candidate, and set  $eval \leftarrow eval + 1$ 
16         add the accepted candidate to  $Q$ 
17         if  $eval \geq MaxEval$  then break end if
18     end for
19     set  $R \leftarrow \text{deduplicate}(P \cup Q)$  and  $P \leftarrow \text{EnvironmentalSelection}(R, ps)$ 
20     let  $A_{old}$  be the signature of  $A$ , then update  $A \leftarrow \text{UpdateArchive}(A, P)$ 
21     if  $\text{Signature}(A) = A_{old}$  then  $stall \leftarrow stall + 1$  else  $stall \leftarrow 0$  end if
22 end while
23 return  $A$ 

```

3.1. Encoding Strategy

A three-layer encoding scheme is used to represent a scheduling solution, including the factory assignment layer (FAL), operation sequence layer (OSL), and machine selection layer (MSL), as shown in Figure 1. FAL represents the factory assigned to each job. Its length equals the number of jobs, and each gene corresponds to one job with a value equal to the factory index. OSL represents the operation scheduling sequence. Its length equals the total number of operations, and repeated job indices are used to represent operations; the t -th occurrence of a job index corresponds to the t -th operation of that job, and the sequence position determines the scheduling priority during decoding. MSL represents machine selection. It is encoded according to the internal operation order of jobs, and each gene corresponds to the selected machine for one operation from its eligible machine set.

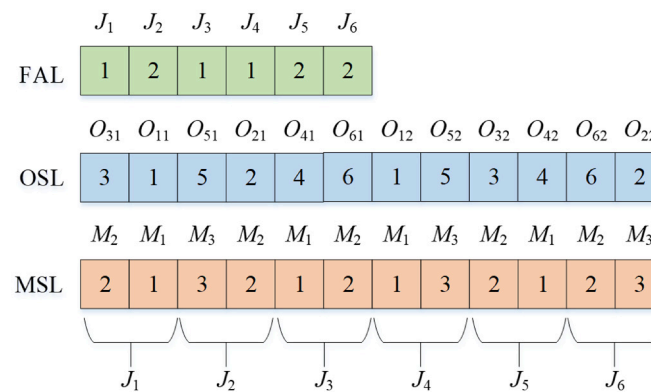


Figure 1. Encoding example.

3.2. Decoding Method

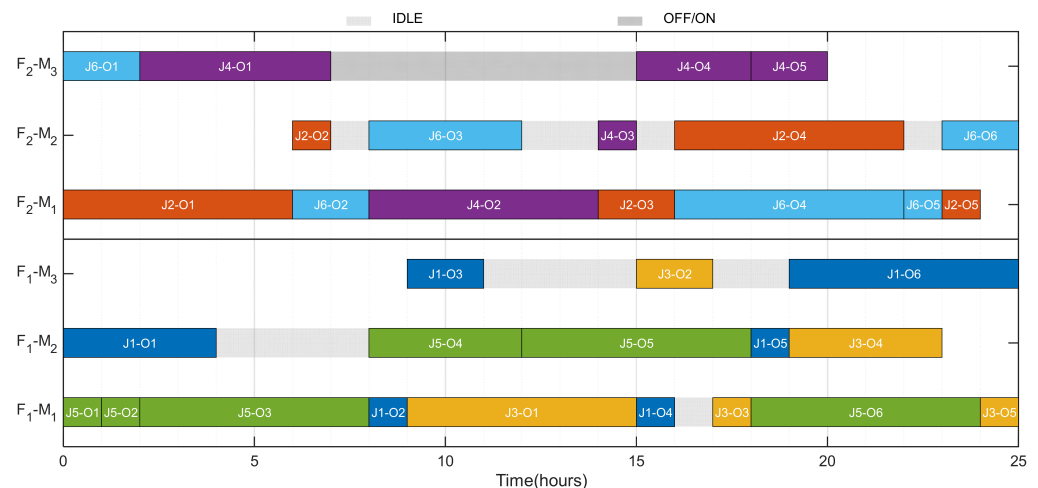
During decoding, the factory assignment layer first determines the factory to which each job belongs, thereby forming factory-level scheduling sets. The operation sequence layer is then scanned to read job indices, and the corresponding operation order is recovered according to the technological route. The machine selection layer subsequently assigns a specific processing machine to each operation. Processing times are determined according to the earliest feasible time rule; the start time of an operation is the larger of the completion time of its predecessor and the earliest available time of the selected machine. Once an operation is scheduled, the earliest available time of the corresponding machine is updated until all operation intervals are obtained. On this basis, idle intervals between adjacent tasks on the same machine are recorded, and on–off states are determined according to the threshold. Internal idle intervals that reach or exceed T_{off} trigger on–off cycles and contribute to on–off energy, whereas idle intervals below the threshold remain in standby mode and contribute to idle energy. The total energy consumption is calculated from processing energy, internal idle standby energy, and on–off cycle energy, and it is optimized together with the makespan.

A small-scale case is used below to illustrate the decoding process, machine on–off judgment, and energy calculation. The case contains two homogeneous factories, denoted as F_1 and F_2 , and each factory is equipped with the same three machines, denoted as M_1 , M_2 , and M_3 . There are six jobs, J_1 – J_6 , and each job contains at most six operations. The processing times of all operations are reported in Table 3, where ‘-’ indicates that the corresponding job cannot be processed on that machine. The energy parameters used in this illustrative example are $P^{proc} = 10$ kW, $P^{idle} = 1.2$ kW, and $e_{sw} = 5$ kWh; therefore, $T_{off} = 5/1.2 = 4.17$ h.

According to the encoding scheme and related constraints, an individual is randomly generated and decoded, producing the Gantt chart shown in Figure 2. The two objective values are $C_{max} = 25$ h and $E_{tot} = 1054.2$ kWh, where $E_{proc} = 1030$ kWh, $E_{idle} = 19.2$ kWh, and $E_{sw} = 5$ kWh. Taking operation J_4 – O_1 as an example, this operation is the first operation of job 4. According to Table 3, it can be processed on M_1 and M_3 , with processing times of 1 h and 5 h, respectively. In the current schedule, it is assigned to M_3 , and therefore, its processing duration is 5 h. In factory F_2 , machine M_3 has an 8-h idle interval in [7, 15] hours. Since the on–off threshold is $T_{off} = 5/1.2 = 4.17$ h, this 8-h idle interval reaches and exceeds the threshold, and therefore one on–off cycle is triggered. On machine M_2 , the intervals [7, 8] hours, [12, 14] hours, [15, 16] hours, and [22, 23] hours are all short idle intervals whose lengths are below 4.17 h; therefore, the machine remains in standby mode and generates idle energy.

Table 3. Processing times of operations (hours).

Jobs	Operation 1			Operation 2			Operation 3			Operation 4			Operation 5			Operation 6		
	M ₁	M ₂	M ₃	M ₁	M ₂	M ₃	M ₁	M ₂	M ₃	M ₁	M ₂	M ₃	M ₁	M ₂	M ₃	M ₁	M ₂	M ₃
J ₁	5	4	-	1	5	3	-	4	2	1	-	5	-	1	-	-	3	6
J ₂	6	-	-	-	1	-	2	-	-	6	6	-	1	-	5	-	-	-
J ₃	6	-	-	-	4	2	1	-	5	6	4	6	1	-	5	-	-	-
J ₄	1	-	5	6	-	-	-	1	-	-	5	-	-	4	2	-	-	-
J ₅	1	5	3	1	-	5	6	-	-	5	4	-	6	6	-	6	4	6
J ₆	-	4	2	2	-	-	-	-	6	6	1	-	-	-	5	3	2	-

**Figure 2.** Gantt chart of the small-scale case.

3.3. Population Initialization

Because DFJSP-OO has a large solution space, complex constraints, and strong interactions between the makespan and energy objectives, a multi-strategy hybrid initialization mechanism is adopted to balance initial solution quality, population diversity, and interpretable scheduling structure. The initial population consists of three types of individuals: random feasible initialization accounts for 40%, and two heuristic initializations based on scheduling rules, each accounts for 30%.

3.3.1. Earliest-Completion Initialization Based on Round-Robin Factory Assignment

FAL initialization adopts a round-robin assignment rule, in which jobs are assigned to factories sequentially according to job indices to reduce the possibility of factory load imbalance in the initial solutions. OSL and MSL are generated jointly using the earliest completion time criterion. Specifically, in each scheduling round, the set of schedulable operations satisfying technological precedence constraints is first constructed. For each candidate operation in this set, all eligible machines are traversed, and the earliest start time and earliest completion time on each machine are calculated. The operation with the smallest earliest completion time is then selected as the current scheduled operation, and its job index is written into OSL. At the same time, the machine that produces the earliest completion time is recorded in the corresponding MSL gene. This process is repeated until all operations are sequenced and assigned to machines.

3.3.2. Minimum-Energy Initialization Based on Round-Robin Factory Assignment

FAL initialization also adopts the round-robin assignment rule to improve the initial distribution of factory loads. OSL and MSL are generated jointly according to the incremental energy criterion. Specifically, in each scheduling round, the set of schedulable operations

satisfying technological precedence constraints is first determined. For each candidate operation, all eligible machines are traversed, and the feasible start time and completion time, and additional energy consumption are calculated according to the current partial schedule. The minimum additional energy of a candidate operation over different machines is then used as its evaluation value. The operation with the smallest evaluation value is written into OSL, and the machine that produces this minimum additional energy is recorded in the corresponding MSL gene. This process is repeated until all operations are sequenced and assigned to machines. Under feasible scheduling constraints, this strategy gives priority to operation-machine combinations with smaller energy increments, which helps improve the energy performance of the initial solutions.

3.3.3. Random Feasible Population Initialization

To improve the diversity of the initial population, a random feasible initialization strategy is introduced. FAL randomly determines the factory assigned to each job. OSL randomly generates an operation sequence under technological precedence constraints. MSL randomly selects a processing machine for each operation from its eligible machine set.

3.4. Evolutionary Operators

DFJSP-OO is characterized by multiple decision layers, strong interactions among variables, and a large solution space. Therefore, hierarchical crossover and mutation mechanisms are used in the design of evolutionary operators to effectively recombine information from different decision layers while maintaining solution feasibility and enhancing global exploration in the objective space.

The factory assignment layer (FAL) and machine selection layer (MSL) both use the uniform crossover (UX) operator. This operator generates a 0–1 mask vector with the same length as the chromosome and exchanges genes at the corresponding positions between two parent individuals, thereby realizing gene-wise recombination for factory assignment or machine selection. Taking MSL as an example, each gene corresponds to the machine selection of a specific operation, and the candidate machine is always restricted to the eligible machine set of that operation. Therefore, feasibility is preserved after crossover without additional repair, as shown in Figure 3.

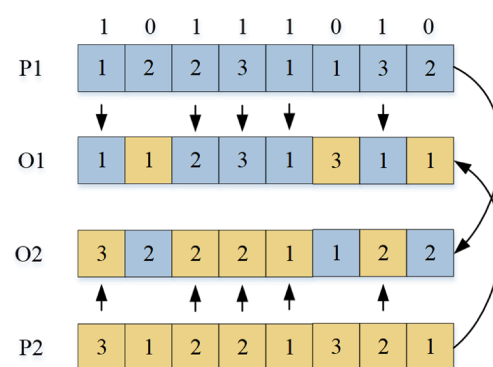


Figure 3. UX in the MSL.

The operation sequence layer (OSL) uses the precedence operation crossover (POX) operator. This operator first randomly divides all jobs into two sets, then inherits the operation sequence structures of the corresponding job sets from the two parents, and finally fills the remaining operations according to their order of occurrence in the other parent. In this way, the scheduling sequence is reconstructed while encoding feasibility is maintained, as shown in Figure 4.

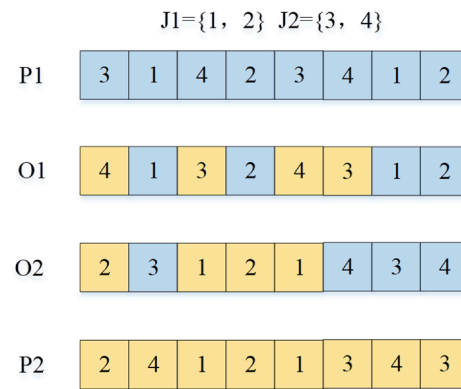


Figure 4. POX in the OSL.

The mutation stage adopts a random perturbation mechanism. In the factory assignment layer, the factory assignments of selected jobs are randomly adjusted. In the operation sequence layer, the operation order is perturbed through random swap or insertion operations. In the machine selection layer, selected operations are reassigned to processing machines from their eligible machine sets.

3.5. Knowledge-Driven Neighborhood Search Operators

The factory assignment, operation sequence, and machine selection decisions in DFJSP-OO are strongly coupled. A single neighborhood structure or purely random perturbation cannot fully exploit the structural information of a solution. Therefore, eight knowledge-driven neighborhood operators are designed around flexible machine selection and critical scheduling structures, as follows.

N1: The operation with the latest completion time in the current schedule is used as the starting point for critical-path backtracking. The current critical path is obtained by tracing technological arcs and machine arcs. The terminal critical operation on this path is then selected, and one machine different from its currently assigned machine is randomly selected from its eligible machine set for replacement.

N2: The factory whose completion time equals the current makespan is defined as the critical factory, and the job with the latest completion time in this factory is selected. A job from another factory is then randomly selected and exchanged with it. After the job exchange, processing machines are randomly reselected from the eligible machine sets for the operations of the migrated jobs.

N3: The machine with the largest current workload is identified, and the operations processed by this machine are extracted. One operation is randomly selected from them, and one machine different from its current machine is randomly selected from its eligible machine set for replacement. If no alternative machine is available for this operation, another eligible operation is selected.

N4: A job is randomly selected from the critical factory and unidirectionally migrated to another factory to reduce the load of the critical factory.

N5: Two operations in the critical factory are randomly selected and their processing order is exchanged, while machine selections remain unchanged.

N6: Two operations in the critical factory are randomly selected, and the latter operation is inserted before the former one, while machine selections remain unchanged.

N7: On the current critical path, several consecutive critical operations processed on the same machine form a critical block. For a head block, an intermediate operation is randomly selected and inserted after the tail operation. For a middle block, two intermediate operations are randomly selected and inserted before the head operation and after the tail

operation, respectively. For a tail block, one operation is randomly selected and inserted before the head operation.

N8: For a critical operation, machine replacement is performed within its eligible machine set. Priority is given to a machine that reduces total energy consumption without worsening the makespan. If no such machine exists, a machine that further shortens the makespan with the smallest increase in energy consumption is selected.

Different neighborhood operators differ in the decision layer, perturbation intensity, and optimization focus. If they are selected in a fixed or uniformly random manner, the search direction may lack specificity, which reduces overall solution efficiency. Therefore, the neighborhood operator is selected according to the structural features of the current schedule. Based on their scope of influence on the scheduling structure, the eight operators are classified into three groups: factory-level reassignment operators (N2 and N4), machine-level adjustment operators (N1, N3, and N8), and operation sequence fine-tuning operators (N5, N6, and N7).

During the search process, the operator category is dynamically determined according to the structural features of the current schedule. When the completion time of the critical factory is higher than that of other factories, factory-level reassignment operators are preferentially invoked to alleviate factory assignment imbalance. When a high-load bottleneck machine appears, machine-level adjustment operators are preferentially invoked for targeted optimization of critical operations or heavily loaded equipment. When the overall system load is relatively balanced, but the makespan remains difficult to reduce, operation sequence fine-tuning operators are preferentially used to refine the critical-path and critical-block structures. After the category is determined, a specific operator within the category is randomly selected to retain search diversity and suppress premature convergence. All neighborhood operators follow a unified acceptance criterion: a new solution is accepted only when it is feasible and is not worse than the current solution in objective values or can contribute a new nondominated solution to the archive.

3.6. Energy-Saving Strategy

In the energy-saving reconstruction stage, once a feasible schedule has been decoded, machine selections and the corresponding processing times remain unchanged. Under this premise, processing energy is fixed, and the energy-saving potential mainly comes from restructuring internal idle periods. Two energy-saving strategies are therefore designed. First, right-shift-based idle reconstruction integrates non-fillable internal idle intervals to form continuous intervals eligible for on–off operation. Second, forward insertion compression further compresses internal idle intervals while maintaining technological feasibility.

3.6.1. Right-Shift-Based Idle Reconstruction Strategy

Without changing the technological precedence relationship within the same job, machine selection, or the makespan, this strategy reconstructs idle structures by controlled right shifts in operations within machine sequences, thereby reducing ineffective idle energy and creating on–off-eligible intervals. The pseudocode is given in Algorithm 2.

This strategy adjusts machine idle intervals without changing job-to-factory assignment, machine selection, or technological precedence. First, operations are right-shifted within their feasible time windows to reduce ineffective internal idle time. Second, adjacent short idle intervals are reorganized when possible so that an internal idle interval can reach the on–off threshold. If the reconstructed schedule does not increase the makespan and reduces total energy consumption, the adjustment is accepted.

Figure 5 shows the Gantt chart obtained after applying the right-shift-based idle reconstruction strategy to the schedule in Figure 2. After the adjustment, the makespan

remains unchanged at $C_{max} = 25$ h, while the total energy consumption decreases from $E_{tot} = 1054.2$ kWh to $E_{tot} = 1041$ kWh, a reduction of 13.2 kWh. Specifically, $E_{proc} = 1030$ and $E_{sw} = 5$ remain unchanged, whereas E_{idle} decreases from 19.2 kWh to 6 kWh; the energy reduction therefore mainly comes from decreased standby time during machine idleness. In detail, M_2 in F_1 delays $J_1 - O_1$ from $[0, 4]$ hours to $[4, 8]$ hours, M_3 in F_1 delays $J_1 - O_3$ to $[13, 15]$ hours, and M_2 in F_2 compresses or eliminates several scattered idle intervals by right-shifting part of the operations. For M_3 in F_2 , the on–off idle interval changes from $[7, 15]$ hours to $[8, 15]$ hours, but the number of on–off cycles does not increase, and the on–off energy remains unchanged. These results show that the right-shift-based idle reconstruction strategy adjusts the idle distribution on machine timelines without increasing the makespan and reduces the standby time counted in energy consumption.

Algorithm 2 Energy-saving reconstruction strategy

Input: Initial feasible schedule S^0 ; on–off threshold T_{off} .

Output: Energy-saving schedule S^1 .

```

1  Set  $S \leftarrow S^0$ ,  $C_{max}^0 \leftarrow f_1(S^0)$ .
2  for each factory  $f = 1, \dots, p$  do
3    for each machine  $k = 1, \dots, m$  do
4      Construct  $\text{Seq}_{fk} = \{a \in \Omega \mid k \in K_a, X_{afk} = 1\}$ , sorted by  $S_a$ .
5      if  $|\text{Seq}_{fk}| \leq 1$ , then continue.
6      Let  $a^{last}$  be the last operation in  $\text{Seq}_{fk}$ .
7      for each operation  $a$  in  $\text{Seq}_{fk}$ , except  $a^{last}$ , do
8        Calculate the maximum feasible right shift  $\xi_a$  under technological
precedence, machine non-overlap, and  $C_{max}^0$ .
9        Generate a temporary schedule  $S'$  by shifting  $a$  rightward by  $\xi_a$ .
10       if  $S'$  is feasible,  $f_1(S') \leq C_{max}^0$ , and  $f_2(S') < f_2(S)$ , then set  $S \leftarrow S'$ .
11     end for
12     Reconstruct  $\text{Seq}_{fk}$  according to the updated start times.
13     for each consecutive triple  $(a^{pre}, a^{cur}, a^{next})$  in  $\text{Seq}_{fk}$  do
14       Compute  $g_1 = S_{a^{cur}} - C_{a^{pre}}$  and  $g_2 = S_{a^{next}} - C_{a^{cur}}$ .
15       if  $g_2 \leq 0$ , then continue.
16       Calculate the maximum feasible right shift  $\xi_{a^{cur}}$  under technological
precedence, machine non-overlap, and  $C_{max}^0$ .
17       if  $g_1 < T_{off}$  and  $g_1 + \xi_{a^{cur}} \geq T_{off}$ , then
18         Set  $\delta = \min\{T_{off} - g_1, \xi_{a^{cur}}\}$ .
19         Generate a temporary schedule  $S'$  by shifting  $a^{cur}$  rightward by  $\delta$ .
20         if  $S'$  is feasible,  $f_1(S') \leq C_{max}^0$ , and  $f_2(S') < f_2(S)$ , then set  $S \leftarrow S'$ .
21       end if
22     end for
23   end for
24 end for
25 Set  $S^1 \leftarrow S$ .
26 return  $S^1$ .

```

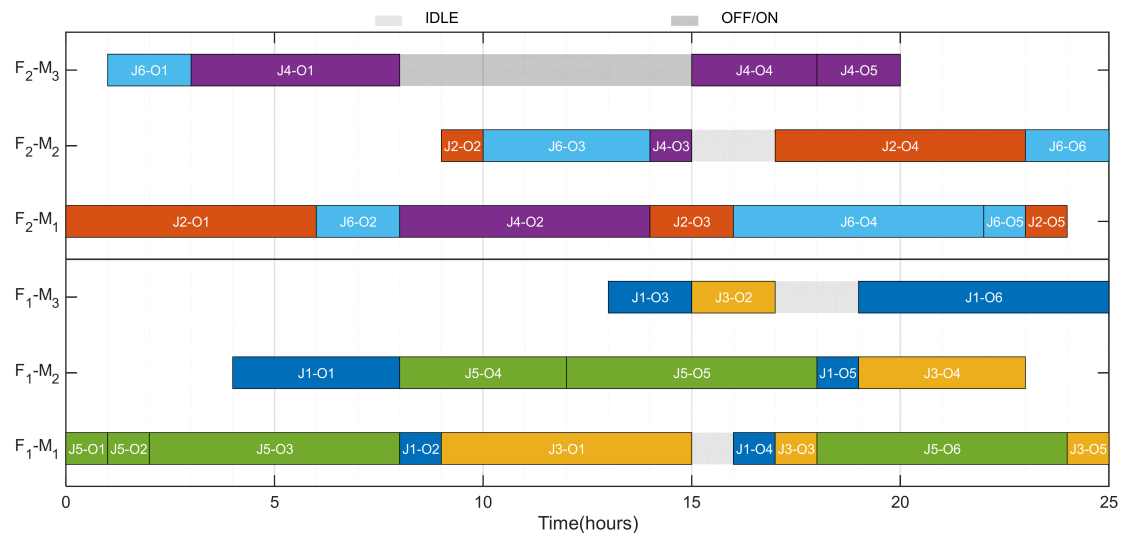


Figure 5. Gantt chart after applying the right-shift-based idle reconstruction strategy.

3.6.2. Forward Insertion Compression Strategy Based on Machine Timelines

This strategy traverses each machine processing timeline and determines whether a subsequent operation can be moved forward into a feasible idle interval before it, while keeping the technological precedence relationship within the same job and other feasibility constraints unchanged. If the idle interval length is no shorter than the operation processing time and the shifted start time is not earlier than the technological ready time of the operation, forward insertion is performed to compress machine idle time and improve the schedule, as shown in Figure 6. In Figure 6, an idle interval [3, 5] hours exists before the operation $J_1 - O_2$, and the processing duration of the operation $J_2 - O_2$ is 1 h. When feasibility conditions are satisfied, $J_2 - O_2$ is moved forward to [3, 4] hours for processing, which partially fills the idle interval, shortens the factory makespan to 6 h, and reduces idle energy by 1 h.

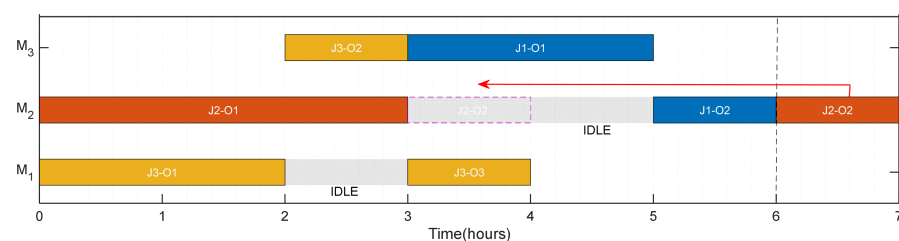


Figure 6. Schematic diagram of the forward insertion compression strategy based on machine timelines.

3.7. Alternative Approaches for Comparison

To provide a broader methodological comparison, three representative multi-objective optimization approaches are considered as alternative approaches: the inverse model and adaptive neighborhood search-based cooperative optimizer (IMANS) [24], the nondominated sorting genetic algorithm II (NSGA-II) [28], and the multi-objective evolutionary algorithm based on decomposition (MOEA/D) [29]. IMANS is a problem-related optimizer developed for energy-efficient distributed flexible job shop scheduling. NSGA-II is a classical dominance-based multi-objective evolutionary algorithm that uses nondominated sorting and crowding distance to balance convergence and diversity. MOEA/D is a decomposition-based multi-objective evolutionary algorithm that transforms a multi-objective problem into a set of scalar subproblems and optimizes them cooperatively. These three methods are selected because they represent problem-specific search, dominance-based optimization, and decomposition-based optimization, respectively.

From the perspective of computational cost, the compared algorithms use the same encoding scheme, decoding rule, crossover-mutation framework, population size, evaluation budget, and number of independent runs. Therefore, solution representation, offspring generation, schedule decoding, objective evaluation, and stochastic-performance evaluation are controlled under the same experimental setting. The main cost differences come from their algorithm-specific search and update mechanisms. NSGA-II mainly relies on nondominated sorting and crowding-distance-based selection; MOEA/D relies on decomposition-based neighborhood update; IMANS includes inverse model and adaptive neighborhood operations; and DKMA introduces knowledge-driven local search and energy-saving reconstruction. The additional cost of DKMA mainly comes from identifying critical scheduling structures and adjusting machine idle intervals. These operations increase the cost of individual refinement, but they support targeted makespan improvement and non-processing energy reduction.

4. Numerical Experiments

The numerical experiments use two benchmark data sets commonly adopted in flexible job shop scheduling, namely Mk01-Mk10 from the Brandimarte benchmark set [30], and 01a-18a from the Dautzère-Pérès benchmark set [31], with 28 instances in total. Each instance is further extended to scenarios with two, three, and four factories, forming the test instance set used to evaluate the proposed algorithm. To assess algorithm performance from different perspectives, two widely used multi-objective optimization metrics are selected: hypervolume (HV) [32] and inverted generational distance (IGD) [33]. HV measures the coverage of the obtained Pareto solution set in the objective space and reflects both convergence and diversity. IGD measures the average distance between the obtained solution set and the reference Pareto front and mainly reflects convergence accuracy. In this paper, both HV and IGD are calculated using normalized objective values to eliminate the influence of different dimensions and numerical scales. Since the true Pareto front is difficult to obtain, all solutions obtained by all algorithms over multiple independent runs are pooled, duplicate solutions are removed, nondominated sorting is performed, and the resulting set is used as the reference set for IGD calculation.

All algorithms are implemented in MATLAB R2016b and run on a computer with an Intel Core i7-6700 CPU @ 3.40 GHz, 16 GB RAM, and Windows 11 operating system. To ensure fair comparison and statistical reliability, each algorithm is independently run 10 times on each test instance, and the mean values of the performance indicators are used to analyze convergence, distribution, and overall solution-set quality. To verify the correctness of the mathematical model and obtain exact solutions for small-scale instances, the MILP model is implemented in Python 3.12.12 and solved by the Gurobi solver.

4.1. Parameter Setting

Algorithm parameters have a considerable influence on solution performance. DKMA contains four key parameters: population size ps , crossover probability pc , mutation probability pm , and local-search proportion pls . To reduce the influence of empirical parameter settings on the experimental results, the Taguchi method is used to analyze these four parameters. Each parameter is assigned four levels: $ps \in \{100, 150, 200, 250\}$, $pc \in \{0.75, 0.80, 0.90, 0.95\}$, $pm \in \{0.05, 0.08, 0.10, 0.12\}$, and $pls \in \{0.10, 0.15, 0.25, 0.35\}$. The detailed combinations are shown in Table 4.

The parameter experiments select three representative instances, Mk03, Mk06, and Mk10, and set 2, 3, and 4 factories for each instance, covering medium-scale, high-flexibility, and large-scale complex scenarios. Under the four-factor, four-level setting, an L16(4⁴) orthogonal array is used to construct the experimental design. Each parameter

combination is independently run 10 times on the nine cases. The termination rule is $MaxEval = 100 \times |\Omega| \times p$, and early termination is allowed when the external nondominated archive is not updated for 10 consecutive generations.

Table 4. Results of the Taguchi orthogonal experiment.

Run	ps	pc	pm	pls	Average HV
1	100	0.75	0.05	0.10	0.741
2	100	0.80	0.08	0.15	0.630
3	100	0.90	0.10	0.25	0.500
4	100	0.95	0.12	0.35	0.658
5	150	0.75	0.08	0.25	0.728
6	150	0.80	0.05	0.35	0.690
7	150	0.90	0.12	0.10	0.433
8	150	0.95	0.10	0.15	0.582
9	200	0.75	0.10	0.35	0.471
10	200	0.80	0.12	0.25	0.438
11	200	0.90	0.05	0.15	0.716
12	200	0.95	0.08	0.10	0.658
13	250	0.75	0.12	0.15	0.353
14	250	0.80	0.10	0.10	0.661
15	250	0.90	0.08	0.35	0.649
16	250	0.95	0.05	0.25	0.670

Using average HV as the main response indicator, the main effects of different parameter levels are compared according to Table 5 and Figure 7. The recommended parameter combination is therefore determined as $ps = 100$, $pc = 0.95$, $pm = 0.05$, and $pls = 0.10$. Since the Taguchi response values were obtained from representative scenarios rather than a single instance, the selected parameter combination reflects an overall response under different problem characteristics. This setting was kept unchanged in all subsequent experiments, and the resulting HV and IGD values suggest that the selected parameter setting is applicable within the tested problem scales.

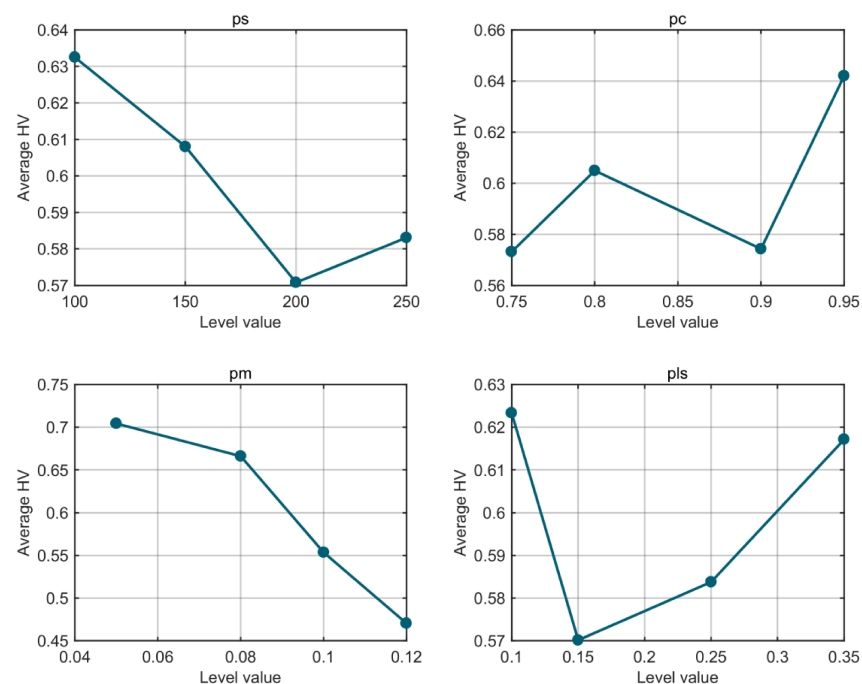


Figure 7. Main-effect response plot of the Taguchi method based on average HV.

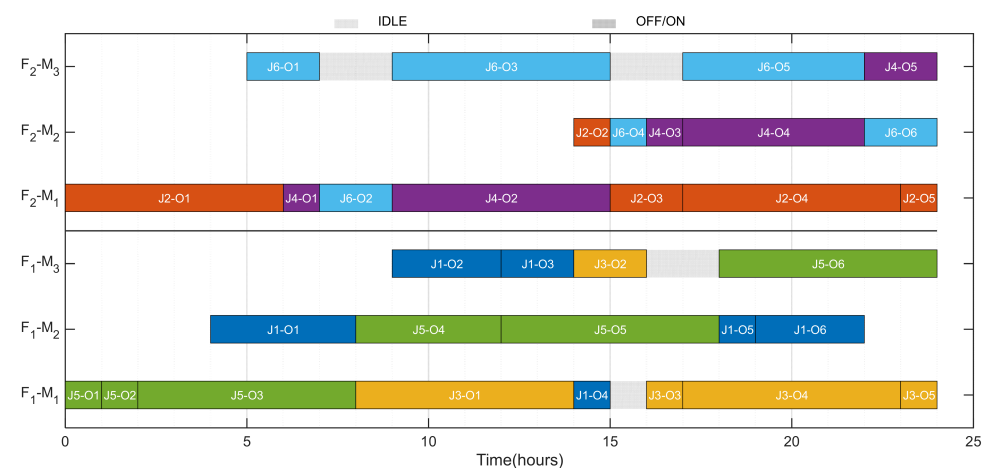
Table 5. Response table of factor main effects based on average HV.

Factor	Level 1	Level 2	Level 3	Level 4	Range Δ	Best Level
ps	0.632	0.608	0.571	0.583	0.062	L1 (100)
pc	0.573	0.605	0.575	0.642	0.069	L4 (0.95)
pm	0.704	0.666	0.554	0.471	0.234	L1 (0.05)
pls	0.623	0.570	0.584	0.617	0.053	L1 (0.1)

4.2. MILP Model Validation

To verify the correctness of the MILP model, the small-scale instance in Table 3 is selected and solved exactly using the Gurobi solver. The computation terminates when the optimality condition is satisfied and the optimality gap is 0. The solution process adopts the ϵ -constraint method and consists of two stages.

In the first stage, the model is solved with the makespan C_{max} as the objective. The result shows that the optimal makespan of this instance is $C_{max} = 24$ h, meaning that the shortest time required to complete all jobs under the current instance and constraints is 24 h. In the second stage, the model is solved again with total energy consumption E_{tot} as the objective under the constraint $C_{max} = 24$ h, thereby obtaining the energy-optimal schedule. The optimal result is $E_{tot} = 1038.4$ kWh, where $E_{proc} = 1030$ kWh, $E_{idle} = 8.4$ kWh, and $E_{sw} = 0$ kWh. The corresponding Gantt chart is shown in Figure 8.

**Figure 8.** Gantt chart of the optimal solution obtained by the Gurobi solver.

The system's energy consumption can be checked from the scheduling result. The total processing time of all operations is 103 h, corresponding to $E_{proc} = 1030$ kWh. Short idle intervals between adjacent processing tasks mainly appear at the following positions: machine M_1 in factory F_1 has two 2-h idle intervals in [7, 9] hours and [15, 17] hours; machine M_1 in factory F_2 has a 1-h idle interval in [15, 16] hours, and machine M_3 has a 2-h idle interval in [16, 18] hours. The total duration of these idle intervals is 7 h, corresponding to $E_{idle} = 8.4$ kWh. None of the idle intervals reaches the machine on-off threshold $T_{off} = 4.17$; therefore, no machine on-off cycle occurs during scheduling, and the corresponding on-off energy is 0 kWh. The resulting total energy consumption is $E_{tot} = 1038.4$ kWh, which is consistent with the MILP solution. This verifies that the proposed MILP model can correctly describe the key scheduling decisions and accurately calculate system energy consumption.

4.3. Ablation Analysis of DKMA Components

The main components of DKMA affect different stages of the search process. The three-layer encoding determines the basic representation of job-to-factory assignment, operation sequencing, and machine selection, and provides the common solution structure

for DKMA and its variants. Hybrid initialization constructs the initial population by combining diversity-oriented random solutions and heuristic solutions with better initial quality. Knowledge-driven local search acts during the iterative search stage and uses critical factories, bottleneck machines, critical paths, and idle intervals to refine candidate schedules. Energy-saving reconstruction acts after decoding and adjusts machine idle structures to reduce non-processing energy. Therefore, the ablation analysis focuses on two removable enhancement components, namely knowledge-driven local search and energy-saving reconstruction, which can be evaluated without changing the basic encoding and decoding framework.

4.3.1. Effectiveness Analysis of the Knowledge-Driven Local Search Strategy

To examine the actual contribution of the knowledge-driven local search strategy to DKMA performance, an ablation experiment is conducted. Keeping population initialization, crossover and mutation, decoding rules, and external archive update unchanged, the knowledge-driven local search strategy is removed from DKMA to obtain the comparison algorithm DKMA-NKLS. Table 6 reports the HV and IGD results of DKMA and DKMA-NKLS on Mk01-Mk10 under different factory configurations. All values are averages over 10 independent runs.

Table 6. Comparison of HV/IGD between DKMA and DKMA-NKLS.

Instances	$n*m$	$p = 2$		$p = 3$		$p = 4$	
		DKMA (HV/IGD)	DKMA-NKLS (HV/IGD)	DKMA (HV/IGD)	DKMA-NKLS (HV/IGD)	DKMA (HV/IGD)	DKMA-NKLS (HV/IGD)
Mk01	10*6	0.733/0.111	0.652/0.130	0.686/0.203	0.675/0.227	0.663/0.187	0.637/0.212
Mk02	10*6	0.692/0.174	0.606/0.219	0.596/0.161	0.468/0.288	0.633/0.133	0.515/0.171
Mk03	15*8	0.598/0.147	0.444/0.203	0.629/0.097	0.658/0.137	0.442/0.111	0.584/0.177
Mk04	15*8	0.554/0.079	0.408/0.128	0.586/0.099	0.523/0.127	0.609/0.102	0.531/0.144
Mk05	15*4	0.604/0.113	0.556/0.129	0.597/0.135	0.567/0.147	0.589/0.113	0.510/0.129
Mk06	10*15	0.341/0.134	0.342/0.147	0.474/0.115	0.428/0.141	0.546/0.166	0.446/0.217
Mk07	20*5	0.495/0.108	0.503/0.188	0.342/0.102	0.364/0.215	0.693/0.096	0.446/0.193
Mk08	20*10	0.636/0.095	0.490/0.135	0.696/0.106	0.722/0.097	0.752/0.122	0.738/0.133
Mk09	20*10	0.441/0.122	0.403/0.134	0.652/0.076	0.396/0.145	0.670/0.097	0.621/0.116
Mk10	20*15	0.451/0.122	0.368/0.158	0.420/0.095	0.375/0.109	0.705/0.088	0.694/0.116
Average		0.555/0.120	0.477/0.157	0.568/0.119	0.518/0.163	0.630/0.121	0.572/0.161

Table 6 shows that DKMA outperforms DKMA-NKLS in most test scenarios. Among the 30 test scenarios, DKMA obtains higher HV values in 24 scenarios and lower IGD values in 29 scenarios. By factory configuration, the average HV values of DKMA under 2, 3, and 4 factories are 0.555, 0.568, and 0.630, respectively, which are higher than the corresponding values of DKMA-NKLS, 0.477, 0.518, and 0.572. The average IGD values are 0.120, 0.119, and 0.121, which are lower than those of DKMA-NKLS, 0.157, 0.163, and 0.161. These results indicate that the knowledge-driven local search improves solution-set coverage and Pareto-front approximation accuracy in most scenarios. A two-sided Wilcoxon signed-rank test is performed on the 30 groups of mean values listed in Table 6. The p -values for HV and IGD are 1.81×10^{-4} and 1.92×10^{-6} , respectively, supporting a statistically significant difference after introducing knowledge-driven local search.

4.3.2. Effectiveness Analysis of the Energy-Saving Strategy

To evaluate the influence of the energy-saving strategy on algorithm performance, another ablation experiment is conducted. With all other mechanisms unchanged, the right-shift reconstruction and forward insertion compression operations are removed from

DKMA to construct the comparison algorithm DKMA-RE. Since this section focuses on differences in total energy consumption between the two algorithms, the relative percentage index (RPI) is used for measurement. A smaller RPI indicates that the energy result obtained by an algorithm is closer to the best value under the corresponding combination. The RPI values in Table 7 are reported as percentages and are defined as

$$RPI_{a,r} = (E_{a,r} - E_{best}) / E_{best} \times 100\%$$

where a denotes the algorithm, r denotes the r -th independent run of that algorithm for a given case, $E_{a,r}$ denotes the total energy consumption obtained in that run, and E_{best} denotes the minimum total energy consumption among the 20 candidate results produced by 10 independent runs of DKMA and 10 independent runs of DKMA-RE.

Table 7. Comparison of RPI (%) between DKMA and DKMA-RE.

Instances	$n*m$	$p = 2$		$p = 3$		$p = 4$	
		DKMA	DKMA-RE	DKMA	DKMA-RE	DKMA	DKMA-RE
Mk01	10*6	0.498	1.100	0.642	1.621	0.834	1.258
Mk02	10*6	0.515	0.960	0.400	0.605	1.114	1.530
Mk03	15*8	0.374	1.137	0.160	1.048	0.172	0.914
Mk04	15*8	0.250	1.692	0.385	1.304	0.045	0.987
Mk05	15*4	0.402	0.496	0.188	0.323	0.240	0.488
Mk06	10*15	0.524	1.989	0.261	1.360	0.229	0.541
Mk07	20*5	0.171	0.702	0.130	0.468	0.293	0.503
Mk08	20*10	0.095	0.376	0.276	0.516	0.104	0.189
Mk09	20*10	0.074	0.204	0.072	0.226	0.056	0.053
Mk10	20*15	0.222	0.333	0.123	0.252	0.121	0.307
Average		0.313	0.899	0.264	0.772	0.321	0.677

Table 7 reports the average RPI comparison between DKMA and DKMA-RE on Mk01-Mk10. DKMA obtains lower RPI values in 29 of the 30 test scenarios. By factory configuration, the average RPI values of DKMA under 2, 3, and 4 factories are 0.313%, 0.264%, and 0.321%, respectively, all lower than the corresponding DKMA-RE values of 0.899%, 0.772%, and 0.677%. The corresponding reductions are 65.23%, 65.87%, and 52.62%. Figure 9 shows the RPI boxplots of the two algorithms over all Mk instances. The two-sided Wilcoxon signed-rank test gives $p = 3.73 \times 10^{-9}$, indicating that DKMA has a statistically significant advantage over DKMA-RE in terms of RPI. Table 7 and Figure 9 jointly show that the energy-saving strategy reduces the average RPI and improves the stability of energy optimization over multiple independent runs.

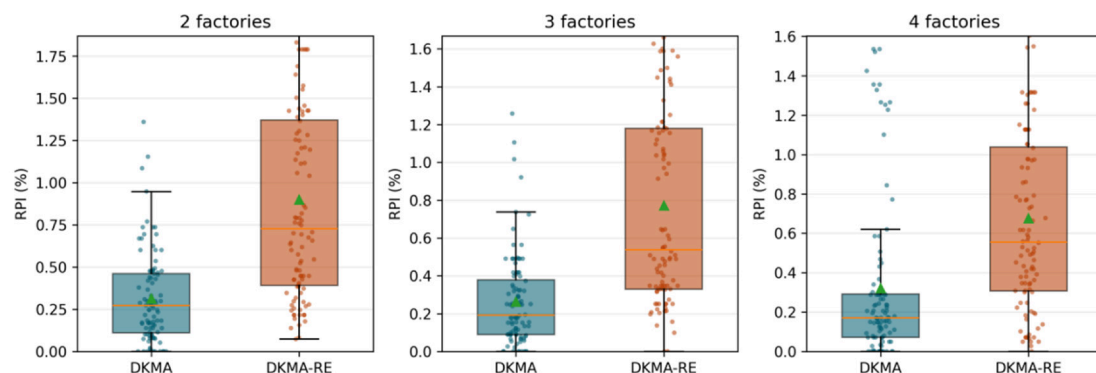


Figure 9. RPI boxplots of DKMA and DKMA-RE.

4.4. Comparative Analysis with Other Algorithms

To evaluate the performance of DKMA, IMANS, NSGA-II, and MOEA/D are used as comparison algorithms. To ensure a fair comparison, all algorithms use the same encoding scheme, decoding process, population size, crossover probability, mutation probability, and other basic parameter settings. The remaining parameters of IMANS follow the original literature. Each instance is independently run 10 times.

Table 8 shows that DKMA obtains the highest HV value in 76 of the 84 test cases. From the average results under different factory scales, the average HV values of DKMA in the 2-, 3-, and 4-factory scenarios are 0.503, 0.528, and 0.522, respectively. These values are higher than those of MOEA/D (0.272, 0.266, and 0.281), NSGA-II (0.274, 0.288, and 0.295), and IMANS (0.270, 0.288, and 0.306). These results indicate that DKMA maintains higher objective-space coverage and overall solution-set quality as the number of factories changes.

Table 8. Normalized HV comparison between DKMA and comparison algorithms on 28 instances.

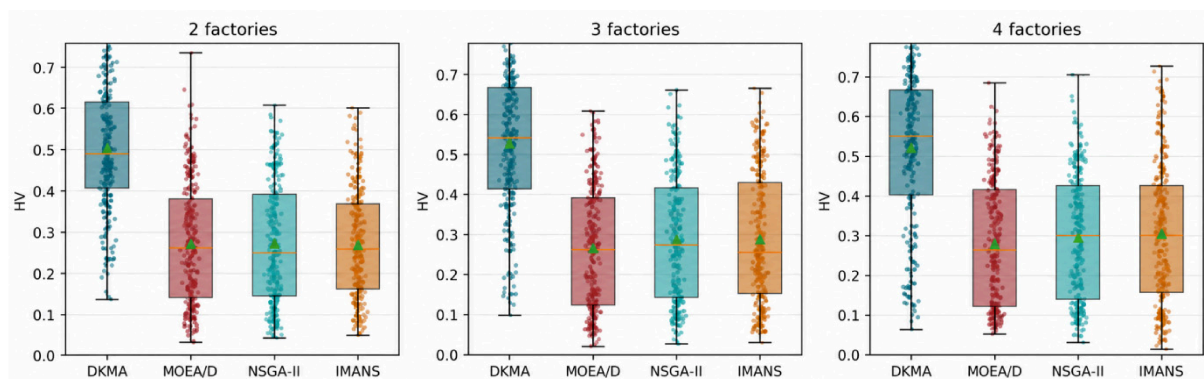
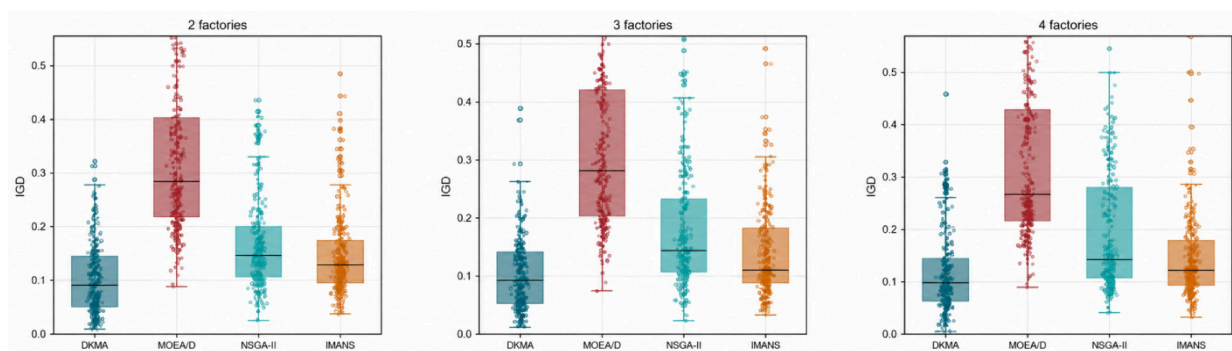
Instances	n^*m	$p = 2$				$p = 3$				$p = 4$			
		DKMA	MOEA/D	NSGA-II	IMANS	DKMA	MOEA/D	NSGA-II	IMANS	DKMA	MOEA/D	NSGA-II	IMANS
Mk01	10*6	0.733	0.306	0.298	0.243	0.686	0.258	0.289	0.199	0.663	0.311	0.364	0.415
Mk02	10*6	0.692	0.133	0.187	0.214	0.596	0.185	0.178	0.144	0.633	0.203	0.148	0.221
Mk03	15*8	0.598	0.127	0.084	0.128	0.629	0.138	0.178	0.185	0.442	0.095	0.126	0.065
Mk04	15*8	0.554	0.298	0.271	0.364	0.586	0.262	0.300	0.221	0.609	0.296	0.262	0.337
Mk05	15*4	0.604	0.309	0.259	0.327	0.617	0.401	0.317	0.338	0.589	0.258	0.376	0.381
Mk06	10*15	0.341	0.077	0.112	0.080	0.474	0.045	0.048	0.088	0.546	0.097	0.078	0.106
Mk07	20*5	0.495	0.152	0.160	0.201	0.342	0.107	0.125	0.167	0.693	0.126	0.134	0.181
Mk08	20*10	0.636	0.256	0.231	0.236	0.696	0.176	0.250	0.204	0.752	0.200	0.254	0.212
Mk09	20*10	0.441	0.114	0.107	0.113	0.652	0.131	0.158	0.186	0.670	0.170	0.121	0.181
Mk10	20*15	0.451	0.184	0.157	0.171	0.420	0.082	0.132	0.103	0.708	0.102	0.093	0.099
01a	10*5	0.430	0.415	0.394	0.364	0.475	0.515	0.520	0.482	0.511	0.473	0.464	0.472
02a	10*5	0.503	0.426	0.354	0.346	0.563	0.456	0.494	0.548	0.483	0.511	0.502	0.574
03a	10*5	0.608	0.450	0.491	0.456	0.671	0.416	0.433	0.416	0.741	0.533	0.521	0.635
04a	10*5	0.751	0.525	0.491	0.514	0.728	0.460	0.558	0.523	0.614	0.555	0.582	0.653
05a	10*5	0.485	0.269	0.306	0.222	0.523	0.239	0.301	0.273	0.559	0.296	0.361	0.323
06a	10*5	0.413	0.262	0.230	0.264	0.442	0.134	0.119	0.209	0.346	0.128	0.237	0.197
07a	15*8	0.493	0.466	0.418	0.388	0.513	0.425	0.406	0.462	0.473	0.428	0.494	0.506
08a	15*8	0.498	0.418	0.464	0.337	0.659	0.446	0.485	0.447	0.659	0.440	0.449	0.452
09a	15*8	0.423	0.265	0.289	0.291	0.399	0.262	0.270	0.305	0.576	0.285	0.285	0.324
10a	15*8	0.699	0.275	0.414	0.331	0.811	0.399	0.463	0.489	0.710	0.425	0.416	0.460
11a	15*8	0.364	0.162	0.167	0.173	0.383	0.183	0.197	0.225	0.204	0.123	0.208	0.190
12a	15*8	0.270	0.092	0.080	0.115	0.192	0.064	0.096	0.075	0.173	0.109	0.189	0.099
13a	20*10	0.478	0.420	0.476	0.398	0.403	0.335	0.392	0.419	0.510	0.415	0.392	0.347
14a	20*10	0.667	0.482	0.491	0.441	0.675	0.484	0.497	0.554	0.647	0.460	0.443	0.415
15a	20*10	0.425	0.186	0.196	0.236	0.535	0.288	0.292	0.269	0.283	0.364	0.348	0.336
16a	20*10	0.515	0.369	0.354	0.396	0.639	0.360	0.378	0.365	0.533	0.301	0.246	0.255
17a	20*10	0.308	0.110	0.119	0.121	0.309	0.111	0.107	0.104	0.156	0.090	0.089	0.040
18a	20*10	0.215	0.076	0.068	0.076	0.155	0.089	0.082	0.074	0.134	0.081	0.089	0.088
Average		0.503	0.272	0.274	0.270	0.528	0.266	0.288	0.288	0.522	0.281	0.295	0.306

Table 9 shows that DKMA obtains the lowest IGD value in 81 of the 84 test cases. Its average IGD values under the 2-, 3-, and 4-factory scenarios are 0.103, 0.101, and 0.109, respectively, all lower than those of the three comparison algorithms. This indicates that the solution sets obtained by DKMA are generally closer to the reference Pareto front.

Figures 10 and 11 show the boxplots of HV and IGD for the four algorithms under different factory scales, respectively. In each subplot, the samples of a single algorithm consist of the results from 10 independent runs on 28 instances under the same factory scale, giving 280 observations in total and reflecting the run-level distribution of algorithm performance. To further test the statistical significance of performance differences, two-sided Wilcoxon signed-rank tests are conducted between DKMA and MOEA/D, NSGA-II, and IMANS using the mean values of the two indicators for each case listed in Tables 8 and 9. The test results show that, for HV, the p -values of DKMA versus MOEA/D, NSGA-II, and IMANS are 4.19×10^{-15} , 7.38×10^{-15} , and 7.51×10^{-15} , respectively. For IGD, the corresponding p -values are 2.45×10^{-15} , 2.36×10^{-15} , and 4.01×10^{-15} .

Table 9. Normalized IGD comparison between DKMA and comparison algorithms on 28 instances.

Instances	n^*m	$p = 2$				$p = 3$				$p = 4$			
		DKMA	MOEA/D	NSGA-II	IMANS	DKMA	MOEA/D	NSGA-II	IMANS	DKMA	MOEA/D	NSGA-II	IMANS
Mk01	10*6	0.123	0.408	0.415	0.461	0.202	0.472	0.437	0.522	0.187	0.458	0.426	0.386
Mk02	10*6	0.172	0.590	0.539	0.510	0.161	0.586	0.586	0.622	0.133	0.540	0.585	0.528
Mk03	15*8	0.147	0.612	0.677	0.611	0.097	0.604	0.574	0.559	0.111	0.622	0.596	0.649
Mk04	15*8	0.079	0.394	0.393	0.340	0.099	0.418	0.393	0.450	0.098	0.405	0.424	0.374
Mk05	15*4	0.112	0.361	0.373	0.342	0.131	0.317	0.360	0.363	0.111	0.387	0.304	0.311
Mk06	10*15	0.134	0.658	0.616	0.650	0.115	0.745	0.746	0.667	0.166	0.682	0.695	0.662
Mk07	20*5	0.108	0.565	0.556	0.518	0.102	0.616	0.598	0.554	0.096	0.580	0.592	0.532
Mk08	20*10	0.088	0.439	0.468	0.454	0.106	0.538	0.485	0.529	0.123	0.514	0.463	0.477
Mk09	20*10	0.122	0.595	0.604	0.595	0.076	0.519	0.517	0.492	0.097	0.534	0.573	0.523
Mk10	20*15	0.122	0.524	0.544	0.530	0.095	0.650	0.585	0.616	0.066	0.618	0.631	0.624
01a	10*5	0.153	0.157	0.169	0.187	0.117	0.130	0.119	0.141	0.111	0.124	0.123	0.111
02a	10*5	0.090	0.164	0.182	0.188	0.109	0.156	0.148	0.132	0.169	0.123	0.149	0.116
03a	10*5	0.178	0.207	0.211	0.232	0.099	0.191	0.173	0.182	0.060	0.168	0.169	0.132
04a	10*5	0.122	0.267	0.294	0.281	0.157	0.289	0.247	0.247	0.232	0.288	0.281	0.219
05a	10*5	0.056	0.322	0.298	0.355	0.059	0.332	0.290	0.298	0.086	0.378	0.339	0.374
06a	10*5	0.048	0.297	0.300	0.289	0.067	0.415	0.424	0.369	0.069	0.464	0.390	0.414
07a	15*8	0.151	0.165	0.179	0.195	0.155	0.170	0.173	0.159	0.120	0.164	0.153	0.153
08a	15*8	0.105	0.191	0.152	0.197	0.102	0.210	0.190	0.198	0.090	0.186	0.160	0.157
09a	15*8	0.065	0.321	0.297	0.292	0.066	0.283	0.272	0.229	0.103	0.314	0.324	0.291
10a	15*8	0.075	0.432	0.332	0.365	0.068	0.291	0.250	0.247	0.184	0.302	0.320	0.285
11a	15*8	0.056	0.389	0.381	0.376	0.061	0.390	0.372	0.364	0.072	0.488	0.425	0.442
12a	15*8	0.042	0.464	0.478	0.437	0.064	0.513	0.484	0.507	0.084	0.500	0.447	0.503
13a	20*10	0.204	0.215	0.193	0.225	0.144	0.175	0.158	0.146	0.125	0.220	0.221	0.253
14a	20*10	0.079	0.199	0.192	0.204	0.086	0.251	0.241	0.209	0.091	0.204	0.215	0.214
15a	20*10	0.059	0.351	0.337	0.295	0.034	0.294	0.291	0.299	0.030	0.269	0.275	0.273
16a	20*10	0.085	0.271	0.266	0.254	0.086	0.324	0.315	0.306	0.087	0.303	0.336	0.329
17a	20*10	0.065	0.474	0.465	0.459	0.062	0.463	0.460	0.467	0.078	0.508	0.516	0.567
18a	20*10	0.051	0.460	0.466	0.462	0.105	0.458	0.457	0.465	0.061	0.552	0.544	0.550
Average		0.103	0.375	0.371	0.368	0.101	0.386	0.369	0.369	0.109	0.389	0.381	0.373

**Figure 10.** HV boxplots of the four algorithms under different factory configurations.**Figure 11.** IGD boxplots of the four algorithms under different factory configurations.

These results show statistically significant differences between DKMA and the three comparison algorithms on the overall test set, indicating that DKMA provides more stable solution-set coverage and Pareto-front approximation on the tested instances.

To further reveal the structural characteristics of nondominated solutions, the results on instance 08a under the 3-factory configuration are selected for case analysis. The Pareto

fronts obtained by the four algorithms are shown in Figure 12. For the solution marked by a star in the DKMA solution set in Figure 12, the objective values are $C_{max} = 1468.0$ h and $E_{tot} = 165,386.6$ kWh. The corresponding Gantt chart is shown in Figure 13.

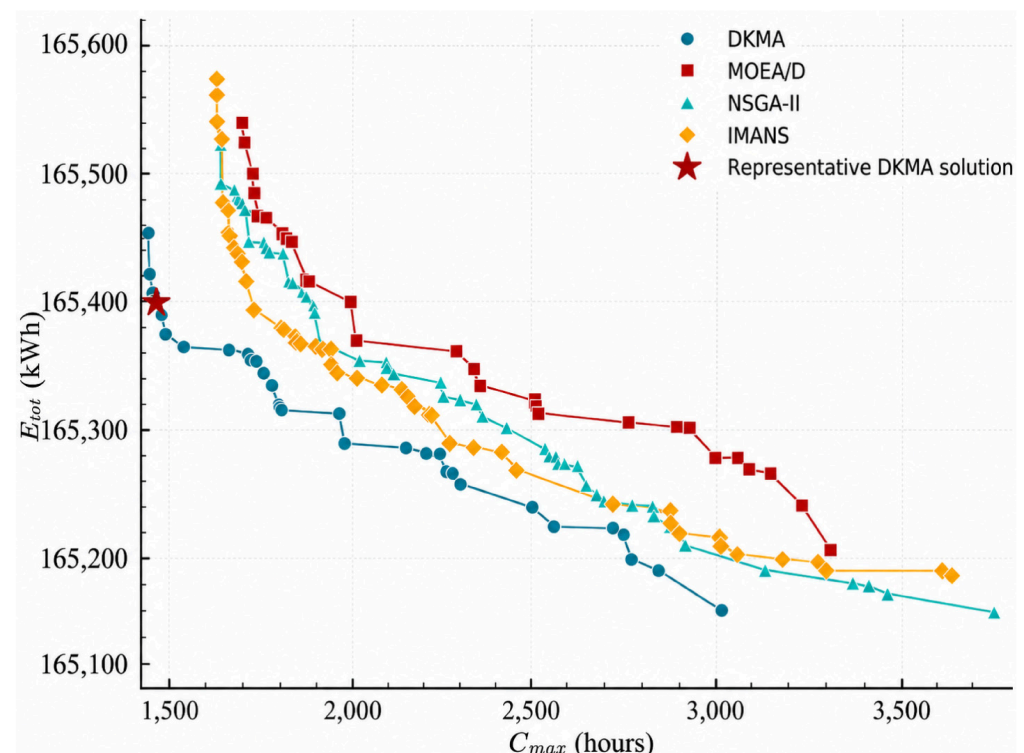


Figure 12. Pareto-front comparison of the four algorithms on instance 08a with three factories.

Figure 13 shows that the completion times of the three factories are 1468 h, 1459 h, and 1454 h, respectively. The critical factory is factory 1, and the maximum completion-time difference among factories is 14 h, indicating a relatively balanced multi-factory task allocation. In terms of energy composition, $E_{proc} = 164,850$ kWh, $E_{idle} = 21.6$ kWh, and $E_{sw} = 515$ kWh. The 515 kWh corresponds to 103 on–off cycles involving all 24 machines used across the three factories. By factory, factories 1, 2, and 3 generate 37, 33, and 33 on–off-eligible idle intervals, respectively. In contrast, only seven short idle intervals fail to reach the on–off threshold, with a total duration of 18 h; therefore, $E_{idle} = 1.2 \times 18 = 21.6$ kWh. Long on–off-eligible intervals appear, for example, in [275, 1100] hours on $F_3 - K_6$, [591, 1137] hours on $F_2 - K_4$, and [212, 696] hours on $F_2 - K_6$, with lengths of 825 h, 546 h, and 484 h, respectively. This representative solution provides a structural explanation of the energy-saving effect: it maintains a low makespan while assigning medium and long idle intervals on multiple machines to the on–off state, thereby reducing standby energy during non-processing periods.

The performance of DKMA is attributable to its use of DFJSP-OO-specific structural information. The algorithm identifies critical positions affecting the makespan and total energy consumption through critical factories, bottleneck machines, critical paths, and on–off-eligible idle intervals, and then uses this information in neighborhood search and energy-saving reconstruction. This mechanism improves solution refinement efficiency and energy optimization capability. The experimental results show that DKMA balances multi-factory load allocation, makespan control, and non-processing energy reduction, supporting the applicability of the proposed method to distributed flexible job shop scheduling with machine on–off energy.

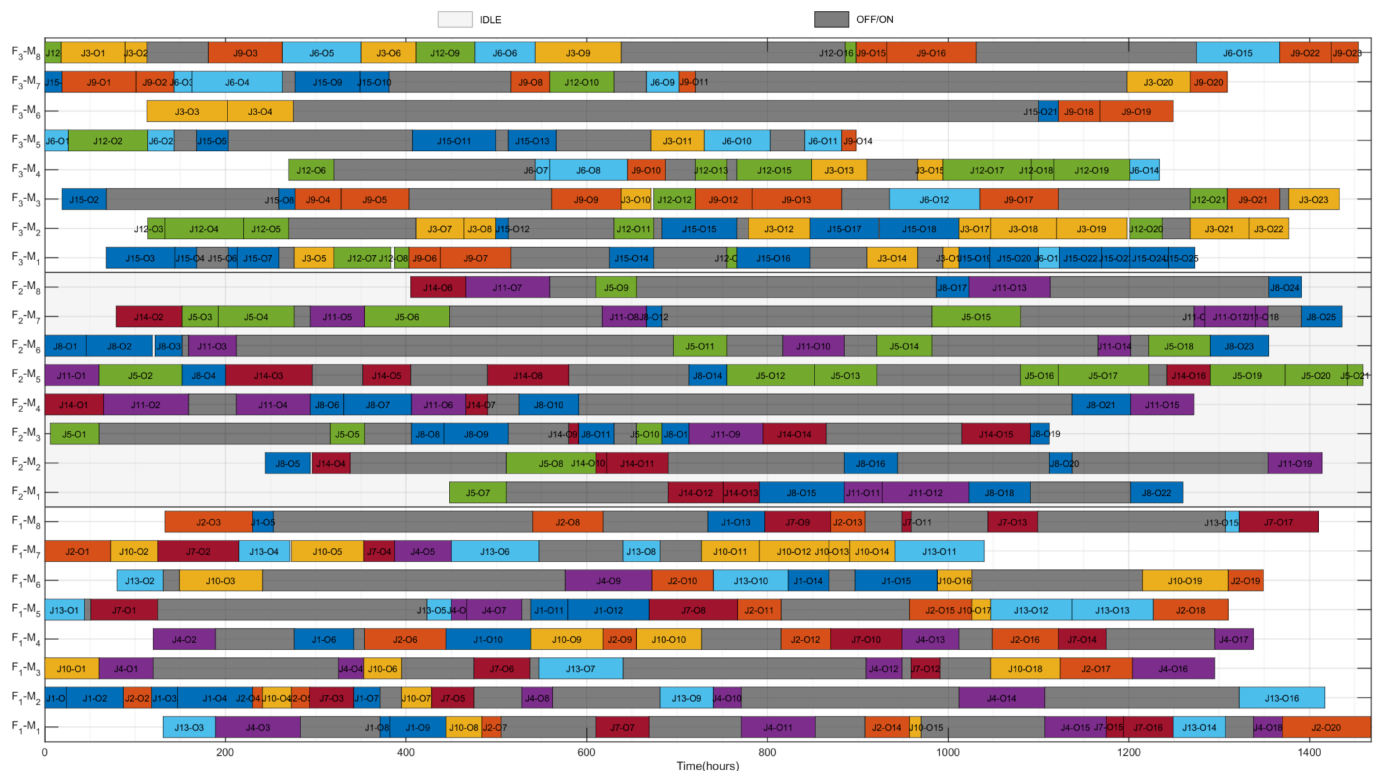


Figure 13. Gantt chart of a representative solution on instance 08a with three factories.

5. Conclusions and Future Work

This paper studies a bi-objective distributed flexible job shop scheduling problem considering machine on–off decisions. A mathematical model is established to jointly account for processing energy, internal idle standby energy, and on–off cycle energy, and a domain-knowledge-driven memetic algorithm is proposed. By considering the relationships among job-to-factory assignment, machine selection, and operation sequencing, the algorithm combines hybrid initialization, knowledge-driven neighborhood search, and energy-saving reconstruction for optimization. Parameter tuning, small-scale MILP validation, ablation experiments, and comparative experiments show that DKMA obtains higher HV and lower IGD values in the tested scenarios.

Future research may extend the present deterministic DFJSP-OO model by incorporating heterogeneous factories, machine breakdowns, dynamic job arrivals, processing-time uncertainty, and maintenance effects. More detailed machine state-transition and energy characteristics, such as setup times, startup delays, sequence-dependent transition constraints, load-dependent power, switching-induced wear, and uncertain energy parameters, can also be considered when production data are available. In addition, acceleration strategies such as parallel decoding and adaptive local-search triggering may improve the applicability of DKMA to larger-scale instances.

Author Contributions: Conceptualization, L.L. and C.G.; methodology, L.L.; software, C.G.; validation, L.L., C.G. and K.G.; formal analysis, C.G.; investigation, L.L.; resources, C.G.; data curation, L.L.; writing—original draft preparation, L.L.; writing—review and editing, C.G.; visualization, K.G.; supervision, L.L.; project administration, K.G.; funding acquisition, K.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the 2026 Henan Provincial Soft Science Research Program, Grant No. 262400410381, and the 2026 Key Scientific Research Project of Colleges and Universities in Henan Province, Grant No. 26A520028.

Data Availability Statement: The datasets used and/or analyzed during the current study are available from the corresponding author on reasonable request.

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

References

1. Naderi, B.; Ruiz, R. The distributed permutation flowshop scheduling problem. *Comput. Oper. Res.* **2010**, *37*, 754–768. [\[CrossRef\]](#)
2. Garey, M.R.; Johnson, D.S.; Sethi, R. The Complexity of Flowshop and Jobshop Scheduling. *Math. Oper. Res.* **1976**, *1*, 117–129. [\[CrossRef\]](#)
3. Dauzère-Pérès, S.; Ding, J.; Shen, L.; Tamssaouet, K. The flexible job shop scheduling problem: A review. *Eur. J. Oper. Res.* **2024**, *314*, 409–432.
4. Wei, G.; Ye, C.; Xu, J. Shared manufacturing-based distributed flexible job shop scheduling with supply-demand matching. *Comput. Ind. Eng.* **2024**, *189*, 109950.
5. Meng, L.; Zhang, C.; Ren, Y.; Zhang, B.; Lv, C. Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem. *Comput. Ind. Eng.* **2020**, *142*, 106347. [\[CrossRef\]](#)
6. Luo, Q.; Deng, Q.; Gong, G.; Zhang, L.; Han, W.; Li, K. An efficient memetic algorithm for distributed flexible job shop scheduling problem with transfers. *Expert Syst. Appl.* **2020**, *160*, 113721. [\[CrossRef\]](#)
7. Zhang, Z.Q.; Wu, F.C.; Qian, B.; Hu, R.; Wang, L.; Jin, H.P. A Q-learning-based hyper-heuristic evolutionary algorithm for the distributed flexible job-shop scheduling problem with crane transportation. *Expert Syst. Appl.* **2023**, *234*, 121050.
8. Tang, J.; Gong, G.; Peng, N.; Zhu, K.; Huang, D.; Luo, Q. An effective memetic algorithm for distributed flexible job shop scheduling problem considering integrated sequencing flexibility. *Expert Syst. Appl.* **2024**, *242*, 122734.
9. Mouzon, G.; Yildirim, M.B.; Twomey, J. Operational methods for minimization of energy consumption of manufacturing equipment. *Int. J. Prod. Res.* **2007**, *45*, 4247–4271. [\[CrossRef\]](#)
10. Fang, K.; Uhan, N.; Zhao, F.; Sutherland, J.W. A new approach to scheduling in manufacturing for power consumption and carbon footprint reduction. *J. Manuf. Syst.* **2011**, *30*, 234–240. [\[CrossRef\]](#)
11. Liu, Y.; Dong, H.; Lohse, N.; Petrovic, S.; Gindy, N. An investigation into minimising total energy consumption and total weighted tardiness in job shops. *J. Clean. Prod.* **2014**, *65*, 87–96. [\[CrossRef\]](#)
12. Dai, M.; Tang, D.; Giret, A.; Salido, M.A.; Li, W.D. Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm. *Robot. Comput.-Integr. Manuf.* **2013**, *29*, 418–429.
13. Gahm, C.; Denz, F.; Dirr, M.; Tuma, A. Energy-efficient scheduling in manufacturing companies: A review and research framework. *Eur. J. Oper. Res.* **2016**, *248*, 744–757. [\[CrossRef\]](#)
14. Fernandes, J.M.R.C.; Homayouni, S.M.; Fontes, D.B.M.M. Energy-Efficient Scheduling in Job Shop Manufacturing Systems: A Literature Review. *Sustainability* **2022**, *14*, 6264. [\[CrossRef\]](#)
15. Shen, L.; Dauzère-Pérès, S.; Maecker, S. Energy cost efficient scheduling in flexible job-shop manufacturing systems. *Eur. J. Oper. Res.* **2023**, *310*, 992–1016. [\[CrossRef\]](#)
16. Park, M.J.; Ham, A. Energy-aware flexible job shop scheduling under time-of-use pricing. *Int. J. Prod. Econ.* **2022**, *248*, 108507.
17. Rui, Z.; Zhang, X.; Liu, M.; Ling, L.; Wang, X.; Liu, C.; Sun, M. Graph reinforcement learning for flexible job shop scheduling under industrial demand response: A production and energy nexus perspective. *Comput. Ind. Eng.* **2024**, *193*, 110325. [\[CrossRef\]](#)
18. Wu, X.; Sun, Y. A green scheduling algorithm for flexible job shop with energy-saving measures. *J. Clean. Prod.* **2018**, *172*, 3249–3264. [\[CrossRef\]](#)
19. Zhang, R.; Chiong, R. Solving the energy-efficient job shop scheduling problem: A multi-objective genetic algorithm with enhanced local search for minimizing the total weighted tardiness and total energy consumption. *J. Clean. Prod.* **2016**, *112*, 3361–3375.
20. Shrouf, F.; Ordieres-Meré, J.; García-Sánchez, A.; Ortega-Mier, M. Optimizing the production scheduling of a single machine to minimize total energy consumption costs. *J. Clean. Prod.* **2014**, *67*, 197–207. [\[CrossRef\]](#)
21. Zhang, Z.; Wu, L.; Peng, T.; Jia, S. An Improved Scheduling Approach for Minimizing Total Energy Consumption and Makespan in a Flexible Job Shop Environment. *Sustainability* **2018**, *11*, 179. [\[CrossRef\]](#)
22. Gu, Y.; Xu, H.; Yang, J.; Li, R. An improved memetic algorithm to solve the energy-efficient distributed flexible job shop scheduling problem with transportation and start-stop constraints. *Math. Biosci. Eng.* **2023**, *20*, 21467–21498. [\[PubMed\]](#)
23. Li, R.; Gong, W.; Wang, L.; Lu, C.; Zhuang, X. Surprisingly Popular-Based Adaptive Memetic Algorithm for Energy-Efficient Distributed Flexible Job Shop Scheduling. *IEEE Trans. Cybern.* **2023**, *53*, 8013–8023. [\[CrossRef\]](#) [\[PubMed\]](#)
24. Cao, S.; Li, R.; Gong, W.; Lu, C. Inverse model and adaptive neighborhood search based cooperative optimizer for energy-efficient distributed flexible job shop scheduling. *Swarm Evol. Comput.* **2023**, *83*, 101419. [\[CrossRef\]](#)
25. Zhou, X.; Wang, F.; Wu, B.; Li, Y.; Shen, N. Deep reinforcement learning-based memetic algorithm for solving dynamic distributed green flexible job shop scheduling problem with finite transportation resources. *Swarm Evol. Comput.* **2025**, *94*, 101885. [\[CrossRef\]](#)

26. Meng, L.; Ren, Y.; Zhang, B.; Li, J.Q.; Sang, H.; Zhang, C. MILP Modeling and Optimization of Energy-Efficient Distributed Flexible Job Shop Scheduling Problem. *IEEE Access* **2020**, *8*, 191191–191203.
27. Wang, J.; Liu, Y.; Ren, S.; Wang, C.; Wang, W. Evolutionary game based real-time scheduling for energy-efficient distributed and flexible job shop. *J. Clean. Prod.* **2021**, *293*, 126093. [[CrossRef](#)]
28. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [[CrossRef](#)]
29. Zhang, Q.; Li, H. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Trans. Evol. Comput.* **2007**, *11*, 712–731. [[CrossRef](#)]
30. Brandimarte, P. Routing and scheduling in a flexible job shop by tabu search. *Ann. Oper. Res.* **1993**, *41*, 157–183. [[CrossRef](#)]
31. Dauzère-Pérès, S.; Paulli, J. An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. *Ann. Oper. Res.* **1997**, *70*, 281–306. [[CrossRef](#)]
32. While, L.; Hingston, P.; Barone, L.; Huband, S. A faster algorithm for calculating hypervolume. *IEEE Trans. Evol. Comput.* **2006**, *10*, 29–38. [[CrossRef](#)]
33. Audet, C.; Bibeon, J.; Cartier, D.; Le Digabel, S.; Salomon, L. Performance indicators in multiobjective optimization. *Eur. J. Oper. Res.* **2021**, *292*, 397–422. [[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.