TSINGHUA SCIENCE AND TECHNOLOGY
DOI: 10.26599/TST.2026.9010043

Imitation Learning and Constraint Programming-assisted Evolutionary Algorithm for Resource-Constrained Flexible Job Shop Scheduling Problem with Sequence-Dependent Setup Times

Weiyao Cheng, Chaoyong Zhang, Leilei Meng*, Hongyan Sang*, Biao Zhang

Abstract: The flexible job shop scheduling problem with sequence-dependent setup times (FJSP-SDST) is a critical issue in intelligent manufacturing industries. Existing research on FJSP-SDST typically assumes that setup times are autonomously managed by machines, which limits its applicability. This paper addresses this gap by exploring the resource-constrained FJSP-SDST (RFJSP-SDST), which considers setup times that are conducted by external resources, such as robots or humans. As an NP-hard problem, it consists of four subproblems: operation sequencing, machine selection and setup task assignment, and resource allocation, making it challenging to solve efficiently. To tackle this complexity, this paper proposes an imitation learning and constraint programming-assisted evolutionary algorithm (ILCPEA) to effectively solve the RFJSP-SDST, focusing on minimizing the makespan. The ILCPEA incorporates a hybrid decoding strategy with combining basic, matheuristic, and imitation learning-assisted methods, which ensures diversity, optimal resource allocation, and a balance between computational resources and performance during the evolution process. To enhance the local search effectively, the disjunctive graph model is used to identify critical paths and four neighborhood structures are designed to improve algorithm convergence. Additionally, a CP-based mathematical evolution operator is introduced to explore the full solution space. Experimental results demonstrate that ILCPEA efficiently generates competitive solutions, outperforms other existing advanced algorithms and demonstrates its practicality in addressing real workshop problems.

Key words: flexible job shop scheduling problem; imitation learning; matheuristic; evolutionary algorithm; sequence-dependent setup time;

1 Introduction

The flexible job shop scheduling problem (FJSP) has been researched in recent years and is widely applied in intelligent manufacturing enterprises [1, 2]. As an extension of the FJSP, the FJSP with sequence-dependent setup times (FJSP-SDST) has attracted growing attention due to its practical application in light industries, such as printing and chemical [3]. However, existing studies of FJSP-SDST generally assume that setup times are managed autonomously by machines, requiring no external intervention. This assumption is limited and only reasonable in a few industries. In many practical production environments, such as automotive engine manufacturing, setup times often are conducted by resources, such as robots or humans [4]. Therefore, this paper further explores resource-constrained FJSP-SDST (RFJSP-SDST) to better reflect real-world conditions, focusing on minimizing the makespan.

As a generalization of the FJSP, the RFJSP-SDST requires decisions on operation sequencing, machine selection, setup task assignment, and resource allocation. Due to its strong NP-hard nature, designing an efficient algorithm for RFJSP-SDST faces three challenges: (1) Unreasonable resource allocation strategy leads to significant wait times for completing setup tasks, thereby increasing the makespan. (2) The critical path in RFJSP-SDST lacks study, makes it challenging to design effective neighborhood structures. (3) The encoding-decoding method imposes limitations on the exploration of the full solution space.

To address these challenges, this paper proposes the imitation learning (IL) and constraint programming (CP) -assisted evolutionary algorithm (ILCPEA). First, we have designed a hybrid decoding strategy to allocate resources effectively, which combines basic,

Weiyao Cheng, Leilei Meng, Biao Zhang and Hongyan Sang are with the School of Computer Science, Liaocheng University, Liaocheng, 252000, China. E-mail: weiyao_cheng@163.com; mengleilei@lcu-cs.com; zhangbiao@lcu-cs.com; sanghongyan@lcu-cs.com.
Chaoyong Zhang is with the School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China. E-mail: zcyhust@hust.edu.cn.
* To whom correspondence should be addressed.

matheuristic, and IL-assisted decoding methods. Specifically, basic decoding provides diversity in resource allocation, matheuristic decoding achieves optimal resource allocation, and IL-assisted decoding balances computational resources and performance in the allocation process. Next, we design the disjunctive graph model for RFJSP-SDST. Based on the disjunctive graph model and identified critical path, four problem-specific neighborhood structures are designed and integrated in local search to enhance convergence. The solution space of RFJSP-SDST may be limited by the encoding-decoding methods. Finally, a CP-based mathematical evolution operator is presented to explore the full solution space. In summary, the following contributions of our paper are summarized as four aspects:

(1) A novel CP model is proposed to achieve optimal solutions for small-scale instances of RFJSP-SDST.

(2) A hybrid decoding strategy is designed to effectively allocate resources, which leverages basic, matheuristic and IL-assisted decoding methods.

(3) A problem-specific local search that incorporates four critical path-based neighborhood structures is presented to further improve the convergence of the algorithm.

(4) To make up for the limitations of the encoding–decoding method, a CP-based mathematical evolution operator is presented to achieve exploration of the full solution space.

The rest of the paper is described as below: Section 2 gives the literature review and research gaps in RFJSP-SDST, IL, matheuristic, and CP-based mathematical evolution. Section 3 presents the definition of RFJSP-SDST and establishes the CP model. Section 4 gives the proposed ILCPEA in detail. Section 5 presents the comparison results and discussion. Section 6 summarizes the work and provides future research directions.

2 Literature Review

2.1 RFJSP-SDST

The FJSP-SDST, which does not consider limited resources for completing setup tasks, is the foundation of RFJSP-SDST. Both problems are NP-hard, and current research mainly relies on exact methods and approximate approaches. The mixed integer linear programming (MILP) and CP models belong to exact methods, which can achieve optimal solutions for small-sized instances. However, for solving large-scale instances, approximate approaches especially metaheuristic algorithms are effective solving tools, leveraging evolution operators and search mechanisms to well explore the solution space. Table 1 summarizes existing papers about FJSP-SDST and RFJSP-SDST. As shown in Table 1, there are five papers with using MILP models, two papers with using CP models, and seven papers with using metaheuristic algorithms.

In summary, only two papers have studied the RFJSP-SDST, and most research overlooks the intervention by resources for setup tasks. To better satisfy real-world conditions, this paper studies the RFJSP-SDST. Additionally, the two papers do not design effective resource allocation strategies, leading to significant wait times. Furthermore, they do not study the critical path in RFJSP-SDST, which reduces the effectiveness of the local search.

Table 1 Existing papers about FJSP-SDST and RFJSP-SDST.

References	Problem	Objectives	Methods
Mousakhani et al. [5]	FJSP-SDST	total tardiness	MILP model and iterated local search
Shen et al. [3]	FJSP-SDST	makespan	MILP model and tabu search (TS) algorithm
Defersha et al. [6]	FJSP-SDST	makespan	parallel genetic algorithm
Winklehner et al. [7]	FJSP-SDST	makespan	CP model
Abdelmaguid et al. [8]	FJSP-SDST	makespan	TS algorithm
Zhang et al. [9]	FJSP-SDST	makespan	MILP and CP models
Kress et al. [4]	RFJSP-SDST	makespan and total tardiness	MILP model and decomposition-based heuristic algorithm
Barak et al. [10]	RFJSP-SDST	operators' idleness cost and setup time-related expenses	MILP model and invasive weed optimization algorithm

2.2 Matheuristic

Matheuristic is a novel optimization paradigm that combines exact mathematical programming techniques with approximate metaheuristic algorithms [11]. The key idea of this approach is to employ mathematical models to further search solutions within dedicated subspaces, thereby enhancing the effectiveness of metaheuristics. Most literature utilizes matheuristic as decoding methods or local search strategies to explore optimal sequencing or assignment for subproblems of a solution. Table 2 summarizes existing studies that use matheuristic in shop scheduling problems. From Table 2, all literatures integrate the MILP model with metaheuristic algorithm and achieves good performance.

However, the MILP model struggles to directly optimize sequencing and assignment when the problem is complex, requiring additional techniques to adjust

model [12]. For example, reference [13] employs a novel branch-and-cut technique to avoid ineffective searches in the MILP model. Reference [14] decomposes the MILP into a linear programming model to reduce the complexity. This motivates us to develop an effective mathematical model that does not require additional adjustments. Additionally, although matheuristic is a powerful optimizer, unfortunately, there is no free lunch. It requires significant computational resources, and as a decoding method, it can hinder diversity.

Table 2 Existing studies about matheuristic.

References	Problem	Mathematical model	Optimizing subproblems	Application
Fanjul-Peyro [15]	Unrelated parallel machine scheduling problem with additional resources	MILP model	operation sequencing	local search
Lin et al. [16]	No-wait flowshop scheduling problem	MILP model	operation sequencing	local search
He et al. [13]	Flowshop group scheduling problem	MILP model	group sequencing	local search
Fan et al. [17]	FJSP with variable lot-sizing	MILP model	lot-sizing assignment	local search
Fan et al. [18]	FJSP with lot-streaming and machine reconfigurations	MILP model	lot-streaming assignment	local search
Hu et al. [14]	Flexible assembly jobs shop scheduling problem (FAJSP)	MILP model	operation sequencing	decoding
Hu et al. [19]	Energy-aware FAJSP	MILP model	operation sequencing	decoding
Hu et al. [20]	FAJSP considering reconfigurable machine	MILP model	machine and auxiliary module assignment	decoding

2.3 Imitation Learning

IL is a significant area of study within reinforcement learning [21]. It can learn expert system behavior patterns in a task and mimic their decision-making processes. With access to expert knowledge, this learning mechanism leads to quick convergence to an effective policy. In recent years, researchers have used IL to address shop scheduling problems, as summarized in Table 3. From the analysis of Table 3, we find that IL only requires a small amount of computational resources to achieve better solutions when accessing high-quality expert knowledge. Thus, the bottleneck of IL's effectiveness lies in obtaining high-quality expert knowledge.

Table 3 Existing studies about IL.

References	Problem	Objectives	Expert system
Ingimundardottir et al. [22]	Job shop problem (JSP)	makespan	optimal solution
Li et al. [23]	Permutation flow shop scheduling problem	makespan	NEH heuristic
Lee et al. [24]	JSP	makespan	optimal solution
Li et al. [25]	Distributed hybrid flow shop problem with family setup time	makespan	metaheuristic algorithms
Echeverria et al. [26]	FJSP	makespan	CP model
Cheng et al. [1]	FJSP with automated guided vehicles	makespan and total energy consumption	local search

2.4 CP-based Mathematical Evolution Operator

The CP-based mathematical evolution operator is a novel optimizer in shop scheduling fields. A high-quality solution produced by the metaheuristic algorithm is used as the initial solution, and the warm-start strategy of CP model solvers is then employed for further solution improvement. The performance of this operator primarily depends on the CP model's strong exploration capability, which helps reduce the limitations imposed by the encoding–decoding scheme in metaheuristic algorithms. Table 4 summarizes existing studies that use CP-based mathematical evolution operator in shop scheduling problems. These studies leverage this operator to update the current best solutions for benchmark instances, and they are recognized as a milestone in the field of shop scheduling. However, the construction of this operator is nontrivial, as it requires addressing two key issues: the development of the CP model and the implementation of the warm-start strategy within the mathematical programming solver.

2.5 Research Gaps

Through a comprehensive review of existing studies and related techniques, several research gaps can be observed as follows: (1) Existing studies do not design effective resource allocation strategies that minimizes wait time. (2) The study of the critical path of RFJSP-SDST is lacking, leading to the absence of an effective local search mechanism. (3) Existing matheuristic relies on the MILP model, but it requires additional techniques to adjust the model in order to achieve better performance. (4) Matheuristic decoding is a strong optimizer but requires large computational resources and hinders resource allocation diversity. (5) IL uses minimal computational resources to achieve better performance, but acquiring high-quality knowledge is difficult. (6) The CP-based mathematical evolution operation is challenging to construct due to the design of the CP model and the implementation of the warm-start strategy within the CP solver.

Table 4 Existing studies about CP-based mathematical evolution operator.

References	Problem	Objective	Method
Beck et al. [27]	JSP	makespan	combining CP and local search algorithm
Heinz et al. [28]	Parallel machine scheduling with sequence-dependent setups and common servers	makespan	combining CP and constructive heuristics
Meng et al. [29]	Distributed FJSP	makespan	hybrid algorithm of genetic algorithm, variable neighborhood search and CP
Kasapidis et al. [30]	FJSP with multiple resource constraints	makespan	CP-based adaptive large neighborhood Search

To tackle existing research gaps given above, this work develops corresponding solution approaches. For tackling research gap (1), we employ a hybrid decoding strategy to effectively allocate resources and reduce wait time. For research gap (2), we model the disjunctive graph for RFJSP-SDST to identify critical path and design four neighborhood structures. For research gap (3), we use the CP model as the mathematical model. Compared with MILP models, CP models show better effectiveness and efficiency in shop scheduling problems [31]. For research gap (4), to reduce computational resources and increase population diversity, we apply a hybrid decoding strategy, which combines basic, matheuristic, and IL-assisted decoding. Specifically, basic decoding provides diversity in resource allocation, matheuristic decoding achieves optimal resource allocation, and IL-assisted decoding balances computational resources and performance in the allocation process. For research gap (5), high-quality knowledge is acquired through matheuristic decoding, which acts as the best expert system by achieving optimal resource allocation. For research gap (6), we successfully develop a CP model for the RFJSP-SDST and implement a warm-start strategy within the *IBM CPLEX* solver.

3 Problem Description

3.1 RFJSP-SDST Definition

The RFJSP-SDST can be defined as below: there are n jobs, m machines and nsR setup resources in the workshop. Each job i includes n_i operations that should be machined with a determined sequence. Each operation $O_{i,j}$ can be processed on a set of alternative machines, but only one machine is selected for processing. When two different operations $O_{i,j}$ and $O_{i',j'}$ are adjacently assigned on machine k, a sequence-dependent setup time $st_{i,j,i',j',k}$ is needed due to adjustment operations. The adjustment operation requires a setup resource. If none setup resource is available at that time, the process must wait. To facilitate describing of the RFJSP-SDST, an illustrative example involving three jobs, three machines, and two setup resources is shown in Fig. 1. The white block represents the wait time for setup resource 1. From time 13 to 16, setup resource 1 is used to conduct the adjustment operations on M1. During this time, the adjustment operations on M2 must wait until time 16. At that point, setup resource 1 has completed the adjustment operations on M1 and can now perform the adjustment operations on M2.

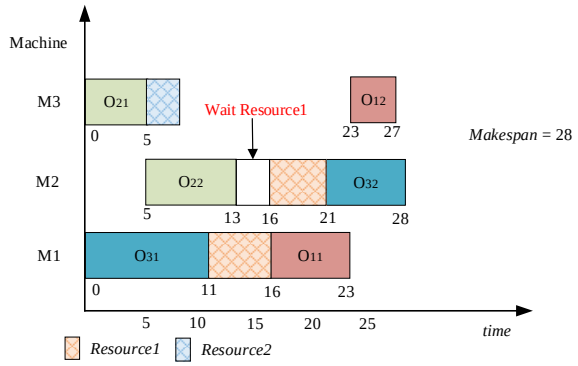

Fig. 1. A representative instance of the RFJSP-SDST.

3.2 Notations

For the CP model, Table 5 gives the parameters definition, and Table 6 presents the decision variables definition.

3.3 CP Model

For the CP model, the objective and constraints are presented as below:

Table 5 CP parameters definition.

Parameters	Definitions
i, i'	Indices for jobs
N_J	Number of jobs
o_i	Total operations number for job i
J	Set of jobs, $J = \{1, 2, 3, ..., N_J\}$
j, j'	Indices for operations
O_i	Operation set of job i, $O_i = \{1, 2, 3, ..., o_i\}$
k	Indices for machines
N_M	Number of machines
N_s	Number of setup resources
M	Set of machines, $M = \{1, 2, 3, ..., N_M\}$
$Op_{i,j}$	The j-th operation for job i
$M_{i,j}$	Candidate machine set for operation $Op_{i,j}$
$pt_{i,j,k}$	Processing time for $Op_{i,j}$ on machine k
$type_{i,j}$	The identifier of $O_{i,j}$
σ	The null identifier
M_{large}	A large positive number
$\tau_{k,type_{i,j},type_{i,j}}$	Sequence-dependent setup time for $O_{i,j}$ and $O_{i',j'}$ is required when they are adjacently processed on machine k
T_k	Transition matrix associated with machine k, incorporating sequence-dependent setup time combinations $\tau_{k,OPType_{i,j},OPType_{i,j}}$.
$C_{\max}$	Makespan

Table 6 CP decision variables definition.

Variables	Definitions
$Proc_{i,j}$	Interval variable for operation $Op_{i,j}$.
$Setup_{i,j}$	Interval variable for setup task of operation $Op_{i,j}$.
$Assign_{i,j,k}$	Optional interval variable associated with the operation $Op_{i,j}$, where the interval size equals $pt_{i,j,k}$. Specifically, machine k is one machine of optional machine set $M_{i,j}$.
Seq_k	Sequence decision variable, and it includes the assigned optional interval variables $mod_{i,j,k}$.

Objective function:

$$\min C_{\max} = \max_{i \in I}(endOf(Proc_{i,o_i})) \quad (1)$$

Constraint sets:

$$lengthOf(Setup_{i,j}) \geq 0, \forall i \in J, j \in O_i \quad (2)$$

$$lengthOf(Setup_{i,j}) = \sum_{k \in K_{i,j}} \tau_{k,type_{i,j},typeOfNext(Seq_k, Assign_{i,j,k}, \sigma)}, \forall i \in J, j \in O_i \quad (3)$$

$$startOf(Setup_{i,j}) \geq endOf(Proc_{i,j}), \forall i \in J, j \in O_i \quad (4)$$

$$endOf(Setup_{i,j}) \leq \sum_{k \in K_{i,j}} startOfNext(Seq_k, Assign_{i,j,k}, M_{large}) \forall i \in J, j \in O_i \quad (5)$$

$$endBeforeStart(Proc_{i,j}, Proc_{i,j+1}), \forall i \in J, j \in \{1, 2, ..., o_i - 1\} \quad (6)$$

$$alternative(Proc_{i,j}, Assign_{i,j,k}), \forall i \in J, j \in O_i, k \in M_{i,j} \quad (7)$$

$$noOverlap(Assign_k, T_k, 1), \forall k \in M_{i,j} \quad (8)$$

$$\sum_{\forall i \in J, j \in O_i} pulse(Setup_{i,j}, 1) \leq N_s \quad (9)$$

where, objective function (1) focuses on minimizing the makespan, and the function $endOf(Proc_{i,o_i})$ returns the ending time of the interval variable $Proc_{i,o_i}$. Constraint set (2) defines that the length of the interval variable $Setup_{i,j}$ is no less than zero. Constraint set (3) indicates that the length of $Setup_{i,j}$ is equal to the sequence-dependent setup time. The function $typeOfNext(Seq_k, Assign_{i,j,k}, \sigma)$ returns the identifier of the next interval variable following the interval variable $Assign_{i,j,k}$ in the sequence Seq_k. If the interval variable $Assign_{i,j,k}$ is the last in the sequence Seq_k, the function returns σ, and $\tau_{k,type_{i,j},type_{i,j}}$ is equal to 0. When the interval variable $Assign_{i,j,k}$ is not present in the sequence Seq_k, the function returns 0, and $\tau_{k,type_{i,j},0}$ is equal to 0. Constraints set (4) indicates that the starting time of the interval variable $Setup_{i,j}$ is greater than or equal to the ending time of the interval variable $Proc_{i,i}$. Constraint set (5) guarantees that the completion time of interval variable $Setup_{i,j}$ does not exceed the start time of the succeeding interval variable after $Assign_{i,j,k}$ in sequence Seq_k. The function $startOfNext(Seq_k, Assign_{i,j,k}, M_{large})$ determines the starting time of the subsequent interval variable. If the interval variable $Assign_{i,j,k}$ is the last in the sequence Seq_k, the function returns M_{large}. When the interval variable $Assign_{i,j,k}$ is not present in the sequence, the function returns 0. Constraint set (6) specifies that interval variable $Proc_{i,i+1}$ can only begin after interval variable $Proc_{i,i}$ has finished. Constraint set (7) specifies that at most one variable $Assign_{i,j,k}$ associated with $Proc_{i,i}$ is present. Constraint set (8) guarantees that the optional interval variables $Assign_{i,j,k}$ cannot overlap in time. Furthermore, the transition matrix T_k imposes a minimum separation between consecutive interval variables, where the separation length is determined by the sequence-dependent setup time. Constraint set (9) restricts the setup resource usage cannot exceed the total number of resources at any time. The function $pulse(Setup_{i,j}, 1)$ returns the cumulative of resource usage.

4 ILCPEA

4.1 ILCPEA Framework

In view of the characteristics of RFJSP-SDST and the shortcomings of current research, this paper introduces an ILCPEA for minimizing the makespan efficiently. The architecture of ILCPEA is outlined as below, and its graphical representation is depicted in Fig. 2.

Step 1: Initialization. Produce *Np* initial individuals randomly and incorporate them into the population.

Step 2: Evolution.

Step 2.1: Crossover. Generate a random value *p* within the range of 0 to 1. If *p* is smaller than the crossover probability *Pc*, select two individuals from the population at random and perform the crossover operators.

Step 2.2: Mutation. Generate a random value *p* within the range of 0 to 1. If *p* is smaller than the mutation probability *Pm*, select an individual from the population at random and perform the mutation operators.

Step 3: Hybrid decoding. Divide the population into three equal subpopulations and apply basic decoding, matheuristic decoding, and IL-assisted decoding respectively.

Step 4: Problem-specific local search. Sort the population and form the elite population with a size *Ne*. Based on the disjunctive graph model, the critical path is first identified for every individual in the elite population, after which a problem-specific local search is carried out with four neighborhoods.

Step 5: CP-based mathematical evolution. If the criterion is reached, the best solution in the population is assigned as the initial solution of the CP model to enhance exploration of the entire solution space. Otherwise, the procedure returns to Step 2. The algorithm terminates its first phase when the algorithm runtime reaches 50% of the predefined timelimit, after which the remaining computational budget is entirely assigned to the CP-based mathematical evolution.

Step 6: Output best solution. After completing the CP-based mathematical evolution, the best solution is output and recorded in the database.

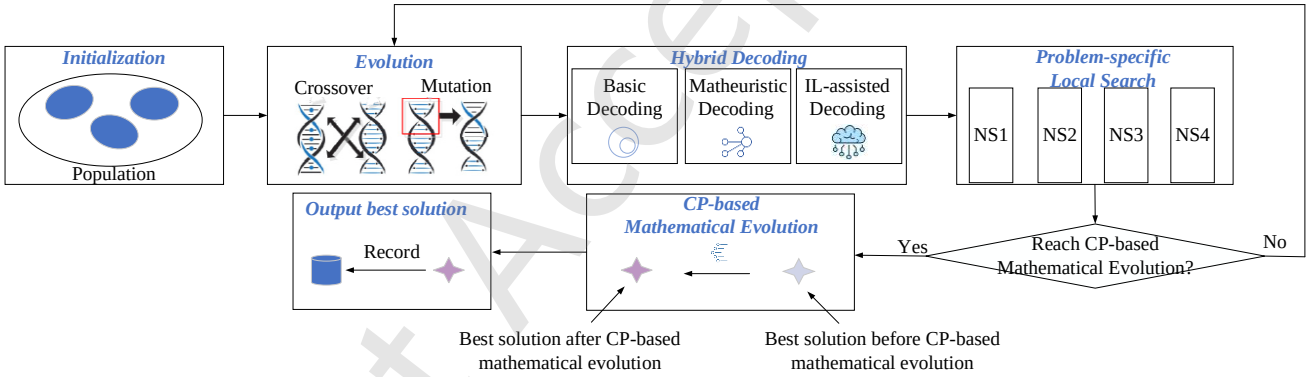

Fig. 2. The framework of ILCPEA.

4.2 Encoding and Evolution Operators

An encoding scheme consisting of three components, namely the operation sequence (OS), machine sequence (MS), and resource sequence (RS). An example of the encoding is given in Fig.3. In the OS representation, each value denotes a job index, while the number of its occurrences corresponds to the operation order of that job. In the MS, the value denotes the processing machine number. In the RS, the value corresponds to the resource number used for tool changes, following a one-to-one mapping with the OS.

Crossover and mutation operators are widely applied in evolutionary algorithms [32, 33]. In this paper, two crossover operators, namely precedence operation crossover (POX) and uniform crossover (UC) are used. POX is applied to the OS, and UC is applied to the MS and RS. For the mutation operators, two types are employed: swapping mutation (SM) and reassigning mutation (RM). SM is applied to the OS, and RM is applied to the MS and RS. More information about these operators can be seen in reference [34].

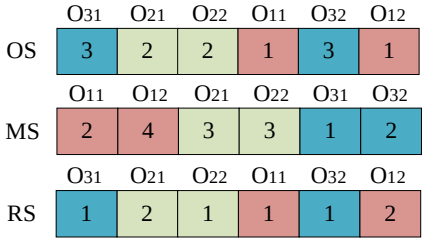

Fig. 3. Example of encoding scheme.

4.3 Hybrid Decoding

To effectively reduce the wait times for completing setup tasks and decrease the makespan, we have designed a hybrid decoding strategy that included basic, matheuristic and IL-assisted decoding. These

combinations enhance population diversity, achieve optimal resource allocation, and balances computational resources and performance.

1) *Basic Decoding*

The idea of the basic decoding process is according to the encoding scheme, akin to living in harmony with nature. This approach prevents the population from converging to the local optimum in the early stages and promotes diversity among the individuals in the population. The pseudocode of basic decoding is presented in Algorithm 1.

2) *Matheuristic Decoding*

Matheuristic decoding is a powerful optimization technique for exploring optimal sequencing or assignment. To achieve optimal resource allocation and avoid the disadvantage of MILP, we utilize the CP model as mathematical model and *IBM CPLEX* solver to implement the matheuristic decoding. The pseudocode of matheuristic decoding is presented in Algorithm 2. Specifically, the matheuristic decoding fixes the OS and MS of the solution as constraints within the CP model, and then inputs the solution into the CP model to find the optimal resource allocation using the *IBM CPLEX* solver.

Algorithm 1 Basic Decoding

Input: OS, MS and RS of an individual
Output: makespan
Begin
For l = 1 to len(OS) do
  Retrieve job index i = OS (l)
  Identify the corresponding operation number j according to the occurrence count of i.
  Select machine k for operation $Op_{i,j}$ according to MS
  Assign resource w for operation $Op_{i,j}$ according to RS
  If machine k has processed earlier operations
    Check if resource w is available
    If resource w is available
      Execute the setup task using the resource w
      Process operation $Op_{i,j}$ on machine k
    Else
      Wait for the resource w until it becomes available
      Perform setup task and process operation
  Else
    Directly process operation $Op_{i,j}$ on machine k
End for
Compute the makespan according to the completion times of the final operations of all jobs.
Return makespan
End

Algorithm 2 Matheuristic Decoding

Input: OS, MS and RS of an individual, Memory vector ($Start$), Memory vector ($Setup\_Start$), Original CP model (CP_o)
Output: makespan
Begin
Basic Decoding (OS, MS, RS)
For l = 1 to len(OS) do
  Retrieve job index i = OS (l)
  Identify the corresponding operation number j according to the occurrence count of i.
  $Start_{i,j}$ ← Starting time of operation $Op_{i,j}$
  $Setup\_Start_{i,j}$ ← Starting time of setup task of operation $Op_{i,j}$
End for
sol = new *IloOplCPSolution*() // Define the initial CP solution sol
For l = 1 to len(OS) do
  Retrieve job index i = OS (l)
  Identify the corresponding operation number j according to the occurrence count of i.
  Select machine k for operation $Op_{i,j}$ according to MS
  $sol.setPresent(Assign_{i,j,k})$ //Machine k is selected for operation $Op_{i,j}$
  $sol.setStart(Proc_{i,j}, Start_{i,j})$ // Starting time of $Op_{i,j}$ is $Start_{i,j}$
  $sol.setStart(Setup_{i,j}, Setup\_Start_{i,j})$ // Starting time of setup task of operation $Op_{i,j}$ is $Setup\_Start_{i,j}$
End for
For l = 1 to len(OS) do
  Retrieve job index i = OS (l)
  Identify the corresponding operation number j according to the occurrence count of i.
  Select machine k for operation $Op_{i,j}$ according to MS
  $CP_o.add(IloPresenceOf(Assign_{i,j,k} = 1))$ // Fix MS in the CP model
  If operation $Op_{i,j}$ is not the first operation of machine k
  $CP_o.add(IloEndBeforeStart(Assign_{i',j',k}, Assign_{i,j,k}))$ // Fix OS in the CP model: operation $Op_{i,j'}$ is processed before $Op_{i,j}$ on machine k
End for
$CP_O.setStartingPoint(\text{sol})$
makespan = $CP_O.getObjValue()$
Return makespan
End

3) *IL-assisted Decoding*

IL is a novel paradigm of reinforcement learning (RL). Different from traditional RL, which requires step-by-step learning from the environment [40], it can acquire knowledge from an expert system for a given task and gradually mimic decision-making processes. By directly learning expert knowledge, this mechanism enables the agent to steadily develop an effective policy and achieve a faster solving speed than the expert system. Due to its advantages, we propose using IL to mimic matheuristic decoding to balance computational resources and performance. The training details and computational overhead analysis of IL are provided in the supplementary file. The expert policy is derived from the matheuristic decoding, which generates expert actions under given OS and MS vectors. For the IL agent, the combined OS and MS vectors are used as inputs.

Additionally, the training dataset includes a total of twenty instances. The pseudocode of IL-assisted decoding is presented in Algorithm 3. The idea of IL-assisted decoding is to use the IL agent to allocate resources based on the solution scheduling state.

Algorithm 3 IL-assisted Decoding

Input: OS, MS and RS of an individual, IL agent (π_{IL}), Memory vector (*Sate*)
Output: makespan
Begin
1: Fill the *Sate* vector with -1
2: For l = 1 to *len*(OS) do
3: Retrieve job index i = OS (l)
4: Identify the corresponding operation number j according to the occurrence count of i.
5: Select machine k for operation $Op_{i,j}$ according to MS
6: Set $Sate_l = i$ and $Sate_{l+len(OS)} = k$
7: Determine resource W for operation $Op_{i,j}$ based on π_{IL}(*State*)
8: If machine k has processed earlier operations
9: Check if resource W is available
10: If resource W is available
11: Execute the setup task using the resource W
12: Process operation $Op_{i,j}$ on machine k
13: Else
14: Wait for the resource W until it becomes available
15: Perform setup task and process operation
16: Else
17: Process operation $Op_{i,j}$ on machine k
18: End for
19: Compute the makespan according to the completion time of the final operation of job.
20: Return makespan
End

4.4 Problem-Specific Local search

1) *Disjunctive Graph Model*

To explore effective neighborhood structures of RFJSP-SDST, the disjunctive graph model $G=(V,C\cup D)$ is employed to identify the critical path. In this model, V represents the set of nodes, and it includes operation task nodes and two fictitious nodes. C denotes the conjunctive arc set, and each arc indicates the sequential relationship between operation tasks. D represents the disjunctive arc set, and each arc connects operation tasks of the same machine in a specific order. Moreover, the arc is used to model setup time in the disjunctive graph. If the resource needs to wait, the wait time is included in the setup time to simplify both expression and calculation. Fig.4 shows a small example of disjunctive graph model. The Start and End nodes are fictitious, while the circle nodes represent operation task nodes. The black solid arrows represent the conjunctive arcs, and the red, blue, and green solid arrows represent the disjunctive arcs.

Specifically, job previous and successor operation task nodes of operation task node u are set as $JP[u]$ and $JS[u]$ respectively, while machine previous and successor operation task nodes are set as $MP[u]$ and $MS[u]$ respectively.

The processing time of operation task node u is $PT[u]$, and the setup time between operation task nodes u and w is $ST[u,w]$. For each operation task node u, it has latest starting time $s^L[u]$ and earliest starting time $s^E[u]$. If $s^L[u]=s^E[u]$, it means the operation task node u is the critical operation. The earliest ending time of node u is denoted by $c^E[u]$, which is calculated as $c^E[u]=PT[u]+s^E[u]$; similarly, the latest ending time of node u is represented by $c^L[u]$, which is calculated as $c^L[u]=PT[u]+s^L[u]$. The critical path is constructed by critical operations in the graph, and the length of path is equal to the makespan [35, 41]. The steps of constructing the critical path are given in Algorithm 4.

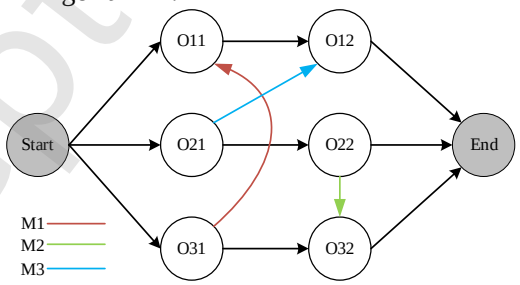

Fig. 4. Illustration of the disjunctive graph model.

2) *Problem-Specific Neighborhood Structures*

Guided by the critical path, four problem-specific neighborhood structures, referred to as NS1, NS2, NS3, and NS4, are constructed. **NS1:** Select an operation of the critical path and pair it with another operation; their execution sequence is then interchanged. **NS2:** Choose a critical-path operation and relocate another operation to a position immediately preceding it. **NS3:** Pick one operation of the critical path together with another operation, and invert the processing sequence of all operations situated between them. **NS4:** Identify an operation on the critical path and reassign it to a different feasible machine selected from its eligible machine set.

Algorithm 4 Critical path construction

Input: critical operation set Ω, disjunctive graph model of a solution
Output: identified critical path
Begin
1: $\Omega = \varnothing$
2: For node u in the node set V:
3: Determine the earliest starting time of node u: $s^E[u]=\max(c^E[MP[u]]+ST[u,MP[u]],c^E[JP[u]])$
4: Determine the latest ending time of node u: $c^L[u]=\min(s^L[MS[u]]-ST[u,MS[u]],s^L[JS[u]])$

5: If $s^L[u] = s^E[u]$:
6: Put node u in critical operation archive, $\Omega \cup u$
7: End for
8: The critical path is generated according to the critical operation set Ω.
9: Return critical path of the solution
End

The four problem-specific neighborhood structures are combined with VNS to form the problem-specific local search strategy, further optimizing elite solutions and reduce the makespan. The pseudocode for this strategy is presented in Algorithm 5. The decoding method for evaluation uses basic decoding.

Algorithm 5 Problem-specific local search
Input: elite population (EP)
Output: new elite population (NEP)
Begin
1: $NEP = \varnothing$
2: For i = 1 to length (EP)
3: Set $x = EP_i$
4: Randomly select a neighborhood structure and perform it on x, set $x' = NS_{random}(x)$
5: Set $l = 1$
6: While $l < 5$
7: Set $x'' = NS_l(x')$
8: If $C_{max}(x'') < C_{max}(x')$
9: Set $x' = x''$ and $l = 1$
10: Else
11: Set $l = l + 1$
12: End while
13: If $C_{max}(x') < C_{max}(x)$
14: $NEP = NEP \cup x'$
15: Else
16: $NEP = NEP \cup x$
17: End for
18: Return new elite population (NEP)
End

4.5 CP-based Mathematical Evolution

The exploration of the full solution space is inherently restricted by the encoding–decoding schemes used in existing approaches and in this study. Although we have implemented matheuristic decoding to achieve optimal resource allocation, there is still a lack of exploration of operation sequencing and machine selection. Specifically, the configuration of operations and machines may not guarantee the scheduling of full solution space.

To overcome these limitations, we designed a CP-based mathematical evolution operator. It uses the obtained high-quality solution as a starting point and then employs the warm-start strategy of the CP solver to further optimize the operation sequence, machine assignment, and resource allocation. By leveraging constraint propagation, domain filtering, and other advanced mechanisms embedded in the CP model, the entire solution space can be effectively explored. The details of the CP-based mathematical evolution are illustrated in Algorithm 6.

Algorithm 6 CP-based Mathematical Evolution
Input: OS, MS and RS of obtained good solution, Memory vector (*Start*), Memory vector (*Setup_Start*), Original CP model (CP_o)
Output: makespan
Begin
1: For l = 1 to len(OS) do
2: Retrieve job index i = OS (l)
3: Identify the corresponding operation number j according to the occurrence count of i.
4: $Start_{i,j}$ ← Starting time of operation $Op_{i,j}$
5: $Setup\_Start_{i,j}$ ← Starting time of setup task of operation $Op_{i,j}$
6: End for
7: opt_sol = new *IloOplCPSolution*()
8: For l = 1 to len(OS) do
9: Retrieve job index i = OS (l)
10: Identify the corresponding operation number j according to the occurrence count of i.
11: Select machine k for operation $Op_{i,j}$ according to MS
12: opt_sol.*setPresent*($Assign_{i,j,k}$)
13: opt_sol.*setStart*($Proc_{i,j}$, $Start_{i,j}$)
14: opt_sol.*setStart*($Setup_{i,j}$, $Setup\_Start_{i,j}$)
15: End for
16: CP_O.*setStartingPoint*(opt_sol)
17: makespan = CP_O.*getObjValue*()
18: Return makespan
End

5 Experimentation

To assess the effectiveness of ILCPEA, a set of comparative experiments are carried out. Both the ILCPEA and the CP model are developed in C++ and run on a workstation with an i7-12700 CPU and an NVIDIA T600 graphics card. The CP model is solved by using CPLEX 12.7.1. Each comparison algorithm is executed 10 times, with the termination condition being when the cumulative CPU running time exceeds twice the total number of operations for the instance. The statistical analysis includes both parametric and non-parametric tests. The paired t-test is employed as the parametric test, while the Wilcoxon signed-rank test and the Friedman test are adopted as non-parametric tests. Because our study involves both pairwise comparisons and multi-algorithm comparisons, different non-parametric tests is required. The Wilcoxon signed-rank test is applied for pairwise comparisons between two algorithms, whereas the Friedman test is used to assess

overall performance differences among multiple algorithms, since it is designed for comparisons involving more than two methods.

5.1 Experiment Settings

The algorithm's performance is evaluated on the widely adopted FJSP benchmark instances MFJS01–10 and MK01–10. The number of available shop-floor resources is fixed at 3 and the sequence-dependent setup time between operations is defined as 30% of the total processing time. The algorithm performance is heavily impacted by the parameters. For the ILCPEA, there have six key parameters: population size N_p, crossover probability P_c, mutation probability P_m, elite population size N_e, learning rate lr and training epoch e. For every parameter, three alternative settings are considered, as detailed below: N_p = {150, 300, 450}, P_c = {0.7, 0.8, 0.9}, P_m = {0.1, 0.15, 0.2}, N_e = {10, 20, 30}, lr = {0.01, 0.05, 0.10} and e = {5, 10, 15}. The Taguchi design of experiments (DOE) is applied to the MK10 instance, with each parameter configuration evaluated based on the relative percentage increase (RPI) [36]. The corresponding formula is provided in the referenced literature [34]. The trend of factor levels is illustrated in Fig.5. Therefore, the parameters are set as below: N_p = 300, P_c = 0.7, P_m = 0.1, N_e = 20, lr = 0.05 and e = 10.

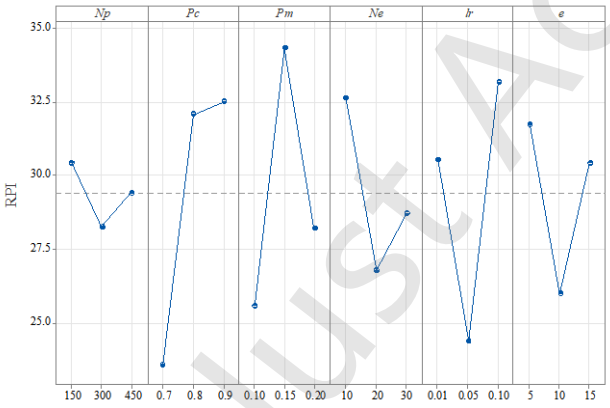

Fig. 5. The factor level trend.

5.2 Model Validation

Table 7 gives the CP model results. For each instance, the timelimit of the CP model is twice the total number of operations. As can be seen from Table 7, the CP model verifies the optimal solutions for MFJS01-03. For MFJS04-MK02 and MK06, the CP model can obtain feasible solutions. Due to the complexity of MK03-05 and MK07-10, the CP model cannot obtain any feasible solutions.

Table 7 The results of CP model.

Instances	Makespan	Time
MFJS01	593	0.04
MFJS02	516	0.07
MFJS03	593	0.07
MFJS04	681	42
MFJS05	653	42
MFJS06	796	48
MFJS07	1157	64
MFJS08	1150	72
MFJS09	1499	88
MFJS10	1658	96
MK01	60	110
MK02	43	116
MK03	-	300
MK04	-	180
MK05	-	212
MK06	111	300
MK07	-	200
MK08	-	450
MK09	-	480
MK10	-	480

5.3 Effectiveness of the Hybrid Decoding

In this section, we will prove the effectiveness of the hybrid decoding strategy compared to four variant algorithms. In Table 8, 'Basic-Algorithm' indicates the use of basic decoding alone, 'Matheuristic' refers to the use of matheuristic decoding alone, 'Basic-Matheuristic' denotes the combination of both basic and matheuristic decoding, and 'Matheuristic-IL' refers to the integration of both matheuristic and IL-assisted decoding. Additionally, 'Best' refers to the best makespan of 10 repeated runs, while 'AVG' represents the average makespan value of 10 repeated runs. The runtime analysis of basic, matheuristic, and IL-assisted decoding are provided in supplemental material.

As reported in Table 8, ILCPEA attains 19 best results in both the Best and AVG metrics across the twenty instances. Moreover, it achieves the lowest average performance values for both indicators. The RPI is used as a performance metric. Fig. 6 presents the boxplots of the RPI of AVG. As indicated, ILCPEA exhibits the best median performance along with smaller deviations. Basic-Algorithm is the worst compared to other algorithms because the resource allocation is unreasonable, leading to increased wait times. For better analysis of the boxplot, we have eliminated the influence of Basic-Algorithm. As shown in Fig. 7, Basic-Matheuristic and Matheuristic-IL are both superior to Matheuristic. The reason is that Matheuristic requires high computation resource, which delays the evolution of the population. Compared to Basic-Matheuristic and Matheuristic-IL, Matheuristic-IL performs better than Basic-Matheuristic in terms of median and exception

values. The reason is that IL-assisted decoding can imitate matheuristic decoding and requires fewer computational resources, while maintaining relatively good allocation and suitable diversity. To provide further evidence of the superiority of the hybrid decoding strategy, the RPI results are examined using both paired t-test and Friedman test, and the outcomes are listed in Tables 9 and 10. In the paired t-test, the p-values are less than 0.05 for Basic-Algorithm, Matheuristic, and Basic-Matheuristic, and equal to 0.05 for Matheuristic-IL. In the Friedman test, the p-value is 0.00. Both results indicate that the hybrid decoding strategy is statistically better than the other decoding methods.

In conclusion, the hybrid decoding strategy effectively allocates resources to minimize wait times by integrating basic, matheuristic and IL-assisted decoding methods. This strategy achieves diversity, optimal resource allocation, and balances computational resources and performance during the evolution process.

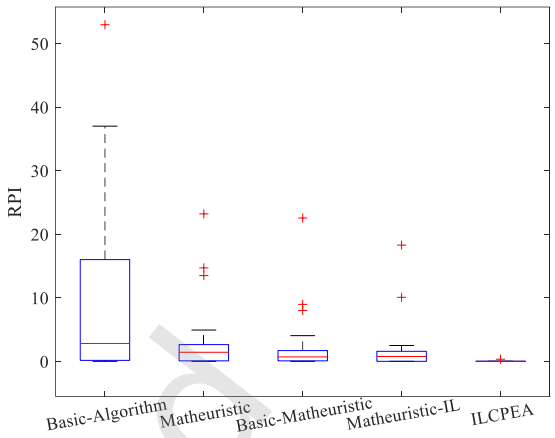

Fig. 6. Boxplots of ILCPEA and other four variant algorithms.

Table 8 Comparison results of ILCPEA and other four variant algorithms.

Instances	Basic-Algorithm		Matheuristic		Basic-Matheuristic		Matheuristic-IL		ILCPEA	
	Best	AVG	Best	AVG	Best	AVG	Best	AVG	Best	AVG
MFJS01	**593**	**593**	**593**	**593**	**593**	**593**	**593**	**593**	**593**	**593**
MFJS02	**516**	**516**	**516**	**516**	**516**	**516**	**516**	**516**	**516**	**516**
MFJS03	**593**	**593**	**593**	**593**	**593**	**593**	**593**	**593**	**593**	**593**
MFJS04	**681**	**681**	**681**	**681**	**681**	**681**	**681**	**681**	**681**	**681**
MFJS05	**653**	**653**	**653**	**653**	**653**	**653**	**653**	**653**	**653**	**653**
MFJS06	**796**	798.4	**796**	797.2	**796**	797.2	**796**	797.1	**796**	**796**
MFJS07	1165	1183.4	1163	1175.9	**1137**	1158.3	**1137**	1165.6	**1137**	**1155.1**
MFJS08	1150	1182.7	1148	1177.3	1150	1175.1	1150	1176.2	**1148**	**1164.7**
MFJS09	1455	1530.1	1461	1510.3	1456	1493.7	1458	1485.9	**1418**	**1473.2**
MFJS10	1658	1697	1651	1700.5	1661	1698.6	1656	1688	**1636**	**1676.9**
MK01	**60**	62.2	**60**	61.2	**60**	60.4	**60**	**60.2**	**60**	60.4
MK02	41	45.2	**40**	41.3	**40**	41	**40**	41.2	**40**	**40.2**
MK03	362	388.6	**325**	327.8	**325**	325.7	**325**	326.6	**325**	**325**
MK04	111	119.2	105	111.9	103	106.5	98	100.5	**95**	**98.6**
MK05	262	268.3	262	268.2	259	265.7	**257**	265	**257**	**264.4**
MK06	114	128.7	102	106.6	101	105.7	96	102.8	**96**	**101.6**
MK07	234	241.1	235	239.5	222	236.8	222	237.8	**219**	**233.7**
MK08	1085	1099.5	1024	1051.8	**983**	1049.1	1027	1053.6	999	**1039.1**
MK09	692	741.7	557	621	546	589.9	552	595.9	**501**	**541.4**
MK10	618	650.1	472	523.6	468.3	520.8	460	502.7	**397**	**425**
Average	642	658.6	621.9	637.5	617.2	633	618.5	631.8	**608**	**621.6**

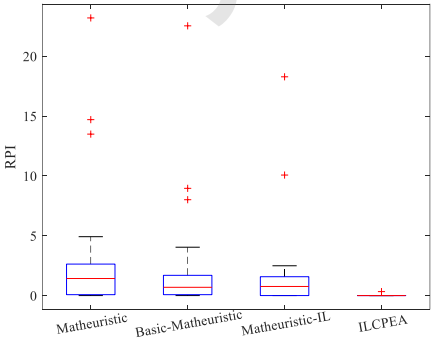

Fig. 7. Boxplots of ILCPEA and other three variant algorithms.

Table 9 Paired t-test for ILCPEA and other four variant algorithms.

Comparisons	p-value	Remark
ILCPEA vs Basic-Algorithm	0.01	<0.05
ILCPEA vs Matheuristic	0.02	<0.05
ILCPEA vs Basic-Matheuristic	0.04	<0.05
ILCPEA vs Matheuristic-IL	0.05	=0.05

Table 10 Friedman test for ILCPEA and other four variant algorithms.

Algorithm	Rank	Min	Max	Mean	Std. Deviation
Basic-Algorithm	4.40	0	52.96	9.634	14.6628
Matheuristic	3.73	0	23.20	3.684	6.1671
Basic-Matheuristic	2.70	0	22.54	2.644	5.3341
Matheuristic-IL	2.60	0	18.28	2.069	4.4104
ILCPEA	1.58	0	0.33	0.017	0.0738
p-value	0.00				

5.4 Effectiveness of the Problem-Specific Local search

This section aims to verify the effectiveness of the proposed problem-specific local search strategy. In Table 11, 'ILCPEA-1' indicates ILCPEA without the problem-specific local search.

As reported in Table 11, ILCPEA attains 20 and 19 best results for the Best and AVG metrics, respectively, across the twenty instances. In addition, compared with ILCPEA-1, it achieves superior overall average performance in both measures. The boxplots of the RPI (AVG) for ILCPEA-1 and ILCPEA are illustrated in Fig. 8, where ILCPEA demonstrates better median performance and smaller variability. The reason is that ILCPEA has four problem-specific neighborhood structures based on the critical path, which further optimize elite solutions and reduce the makespan compared to ILCPEA-1. To further illustrate the effectiveness of the problem-specific local search, a paired t-test and a Wilcoxon signed-rank are conducted on the RPI results. The p-value is 0.02 for the paired t-test and 0.00 for the Wilcoxon signed-rank, both of which are less than 0.05, indicating that the problem-specific local search component is effective. The detailed results of the Wilcoxon signed-rank are shown in Table 12.

In summary, the critical-path-based problem-specific local search utilizes tailored neighborhood structures to refine elite solutions, thereby improving search efficiency and accelerating convergence.

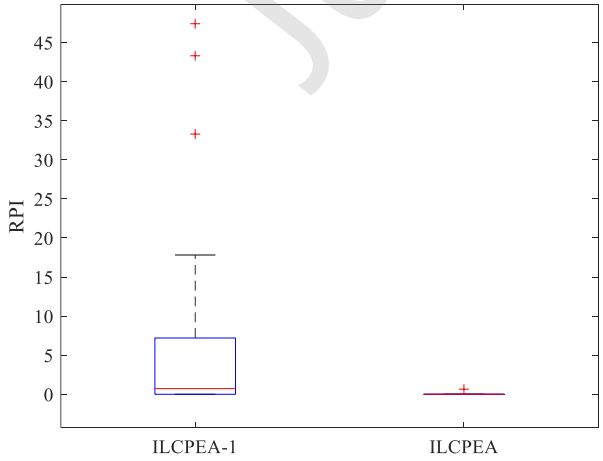

Fig. 8. Boxplots of ILCPEA-1 and ILCPEA.

Table 11 Comparison results of ILCPEA-1 and ILCPEA.

Instances	ILCPEA-1		ILCPEA	
	Best	AVG	Best	AVG
MFJS01	**593**	**593**	**593**	**593**
MFJS02	**516**	**516**	**516**	**516**
MFJS03	**593**	**593**	**593**	**593**
MFJS04	**681**	**681**	**681**	**681**
MFJS05	**653**	**653**	**653**	**653**
MFJS06	**796**	799.5	**796**	**796**
MFJS07	**1137**	1163.6	**1137**	**1155.1**
MFJS08	1149	1170.8	**1148**	**1164.7**
MFJS09	1471	1502.7	**1418**	**1473.2**
MFJS10	1647	1679.2	**1636**	**1676.9**
MK01	**60**	61.2	**60**	**60.4**
MK02	41	43.6	**40**	**40.2**
MK03	**325**	344.3	**325**	**325**
MK04	**95**	99.3	**95**	**98.6**
MK05	258	**262.7**	**257**	264.4
MK06	104	145.6	**96**	**101.6**
MK07	237	344.5	**219**	**233.7**
MK08	1046	1085.1	**999**	**1039.1**
MK09	537	637.9	**501**	**541.4**
MK10	504	566.5	**397**	**425**
Average	622.2	647.1	**608**	**621.6**

Table 12 Wilcoxon signed-rank for ILCPEA-1 and ILCPEA.

Algorithm	Min	Max	Mean	Std. Deviation
ILCPEA-1	0	47.41	8.327	15.0375
ILCPEA	0	0.65	0.033	0.1453
p-value	0.00			

5.5 Effectiveness of the CP-based Mathematical Evolution

This section evaluates the contribution of the CP-based mathematical evolution component. In Table 13, 'ILCPEA-2' denotes the variant of ILCPEA where the CP-based mathematical evolution mechanism is removed.

As reported in Table 13, ILCPEA achieves 20 best results for both Best and AVG across the twenty instances. Moreover, it consistently outperforms ILCPEA-2 in terms of the overall average performance for these two metrics. The boxplots of RPI (AVG) for ILCPEA-2 and ILCPEA are illustrated in Fig. 9, where ILCPEA exhibits superior median performance along with smaller variability. The observed performance gain results from the CP-based mathematical evolution operator, which strengthens the search process by taking advantage of the exploration capability of the CP solver. To provide further evidence of the superiority of the CP-based mathematical evolution, the RPI results are examined using both paired t-test and Wilcoxon signed-rank. The p-value is 0.00 for the paired t-test and 0.00 for the Wilcoxon signed-rank, indicating the effectiveness of the CP-based mathematical evolution component. The detailed results of the Wilcoxon signed-rank are shown in Table 14.

Overall, the CP-based mathematical evolution can make up for the limitations of the encoding–decoding strategy. By exploiting the capabilities of the mathematical solver, including constraint propagation, domain filtering, and related advanced techniques, high-quality solutions can be effectively obtained.

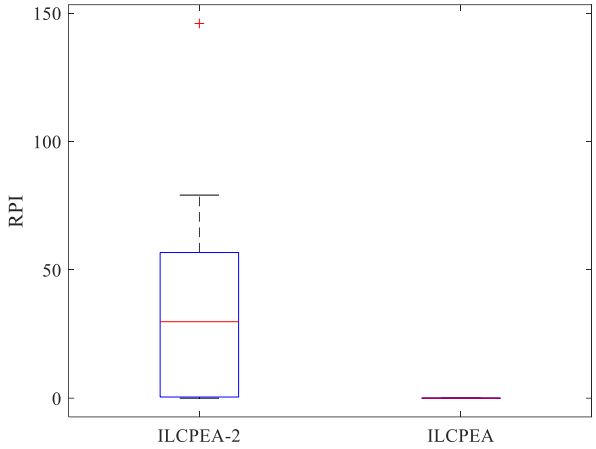

Fig. 9. Boxplots of ILCPEA-2 and ILCPEA.

Table 13 The comparison results of ILCPEA-2 and ILCPEA.

Instances	ILCPEA-2		ILCPEA	
	Best	AVG	Best	AVG
MFJS01	**593**	**593**	**593**	**593**
MFJS02	**516**	**516**	**516**	**516**
MFJS03	**593**	**593**	**593**	**593**
MFJS04	**681**	**681**	**681**	**681**
MFJS05	**653**	655.3	**653**	**653**
MFJS06	807	815.2	**796**	**796**
MFJS07	1255	1395.7	**1137**	**1155.1**
MFJS08	1288	1474.1	**1148**	**1164.7**
MFJS09	1829	1945.7	**1418**	**1473.2**
MFJS10	2019	2191.4	**1636**	**1676.9**
MK01	75	81.1	**60**	**60.4**
MK02	62	72	**40**	**40.2**
MK03	469	490.3	**325**	**325**
MK04	139	145.6	**95**	**98.6**
MK05	330	340.8	**257**	**264.4**
MK06	245	249.9	**96**	**101.6**
MK07	369	381.5	**219**	**233.7**
MK08	1014	1044.8	**999**	**1039.1**
MK09	855	880.4	**501**	**541.4**
MK10	717	756.4	**397**	**425**
Average	725.5	765.2	**608**	**621.6**

Table 14 Wilcoxon signed-rank FOR ILCPEA-2 and ILCPEA.

Algorithm	Min	Max	Mean	Std. Deviation
ILCPEA-2	0	145.96	35.203	37.5306
ILCPEA	0	0	0	0
p-value	0.00			

5.6 Comparison with Other Algorithms

This section provides the comparisons of ILCPEA with several representative state-of-the-art algorithms, including QABC [37], IGA [38], CVNS-Q [39] and MFLA-MH [42]. These methods have demonstrated strong performance in solving extended variants of the FJSP. The algorithm parameters are tuned based on the DQE testing procedure. For QABC, the learning rate and discount factor are set as 0.2 and 0.1 respectively, while the population size and maximum number of iterations are set as 80 and 10 respectively. For IGA, the crossover and mutation probabilities are set as 0.9 and 0.1 respectively, with the population size configured as 400 and the diversity check parameter set to 300. For CVNS-Q, the restart interval and CVNS-Q search duration are set as 200 and 20 respectively; meanwhile, the learning rate and discount factor are set to 0.2 and 0.7 respectively, and the VNS search time is 8. Regarding MFLA-MH, each individual conducts 80 search operations, and the number of local search iterations is determined as 16.

As reported in Table 15, ILCPEA obtains 19 and 18 best results in terms of Best and AVG respectively, across the twenty benchmark instances. Moreover, it achieves superior overall average performance compared with the competing algorithms under both evaluation metrics. Fig. 10 illustrates the boxplots of RPI for AVG, where ILCPEA shows a clearly separated distribution without overlap with the others. In addition, it demonstrates improved median performance together with reduced variability. To further confirm the performance advantage of ILCPEA, statistical analyses using the paired t-test and Friedman test are conducted on the RPI results, with detailed outcomes provided in Tables 16 and 17. The obtained p-values are all below 0.05, verifying that ILCPEA significantly outperforms the compared algorithms from a statistical perspective.

In conclusion, ILCPEA integrates effective components, such as the hybrid decoding strategy, problem-specific local search, and CP-based mathematical evolution, which collectively drive its superior performance.

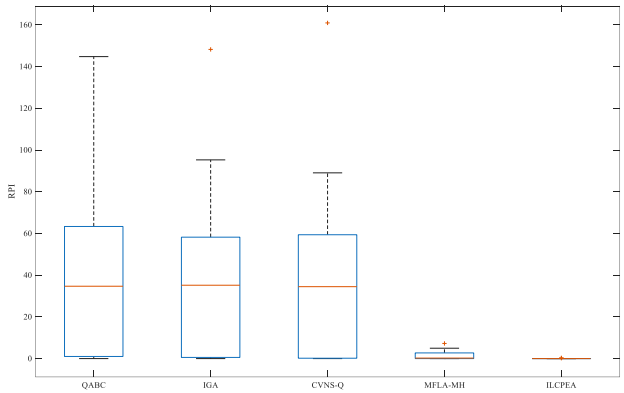

Fig. 10. Boxplots of QABC, IGA, CVNS-Q, MFLA-MH and ILCPEA.

Table 15 The comparison results of QABC, IGA, CVNS-Q, MFLA-MH and ILCPEA.

Instances	QABC		IGA		CVNS-Q		MFLA-MH		ILCPEA	
	Best	AVG	Best	AVG	Best	AVG	Best	AVG	Best	AVG
MFJS01	**593**	**593**	**593**	**593**	**593**	**593**	**593**	**593**	**593**	**593**
MFJS02	**516**	**516**	**516**	**516**	**516**	**516**	**516**	**516**	**516**	**516**
MFJS03	**593**	**593**	**593**	**593**	**593**	**593**	**593**	**593**	**593**	**593**
MFJS04	**681**	**681**	**681**	**681**	**681**	**681**	**681**	**681**	**681**	**681**
MFJS05	**653**	653.8	**653**	**653**	653	655.3	**653**	**653**	**653**	**653**
MFJS06	829	860.5	807	846.7	807	853.3	**796**	**796**	**796**	**796**
MFJS07	1317	1428	1337	1392.8	1358	1430.5	**1137**	1162	**1137**	**1155.1**
MFJS08	1256	1550.6	1470	1560.6	1500	1573.1	**1148**	1166.4	**1148**	**1164.7**
MFJS09	1915	2007.3	1924	2009.8	1866	1986.6	1431	1477.3	**1418**	**1473.2**
MFJS10	2250	2327	2065	2286.4	2153	2277.7	1641	1680.3	**1636**	**1676.9**
MK01	74	79.1	78	82.9	74	81	**60**	63.4	**60**	**60.4**
MK02	69	79.5	72	78.5	67	76	**40**	42	**40**	**40.2**
MK03	473	494.3	479	505.6	481	509	**325**	**325**	**325**	**325**
MK04	151	159	128	141.2	128	141.3	**95**	**98.3**	**95**	98.6
MK05	353	366.2	329	347	327	349	258	268.2	**257**	**264.4**
MK06	229	248.7	244	252.2	256	265.1	99	104.8	**96**	**101.6**
MK07	352	397.1	362	375.9	342	378.9	**219**	244.5	**219**	**233.7**
MK08	1022	1056.4	1029	1048.1	**982**	**1036.6**	1099	1111.9	999	1039.1
MK09	865	892.9	847	891.6	898	924.9	532	553.8	**501**	**541.4**
MK10	716	762	733	776.1	705	753	398	432.6	**397**	**425**
Average	745.4	787.3	747.0	781.6	749.0	783.7	615.7	628.1	**608**	**621.6**

Table 16 Paired t-test for QABC, IGA, CVNS-Q, MFLA-MH and ILCPEA.

Comparisons	p-value	Remark
ILCPEA vs QABC	0.00	<0.05
ILCPEA vs IGA	0.00	<0.05
ILCPEA vs CVNS-Q	0.00	<0.05
ILCPEA vs MFLA-MH	0.01	<0.05

Table 17 Friedman test for QABC, IGA, CVNS-Q, MFLA-MH and ILCPEA.

Algorithm	Rank	Min	Max	Mean	Std. Deviation
QABC	3.90	0	144.78	39.094	38.9679
IGA	3.50	0	148.23	37.708	39.0315
CVNS-Q	3.75	0	160.93	38.186	40.4924
MFLA-MH	2.25	0	7.26	1.562	2.1885
ILCPEA	1.60	0	0.31	0.028	0.0854
p-value	0.00				

5.7 Case Study and Managerial Insights

To assess its effectiveness in industrial settings, ILCPEA is applied to a real-world engine production workshop case. The layout of the workshop and the details of the case are provided in the supplemental material.

In the engine production workshop scheduling system, operation sequencing uses the first-served rule, machine selection uses the load balancing rule, and resource allocation uses the even distribution rule. Table 18 shows the comparison results between the production workshop and ILCPEA. ILCPEA demonstrates the best performance of all four metrics, validating its effectiveness when applied to real-world production.

Based on the analysis of Table 18 data, we have identified two key insights for this production workshop: (1) managers should minimize the number of waits and increase resources as the economy permits. This approach will enhance machine utilization and improve production efficiency. (2) managers should consider multiple metrics when determining operational sequencing, machine selection, and resource allocation rules. For example, when determining operation processing, both the average machine utilization and the number of waits should be considered.

Table 18 Comparison results of production workshop and ILCPEA.

Performance metrics	Production workshop	ILCPEA
C_{max}	296	136
Average processing time of machine	269.67	121.67
Average utilization of machine	18%	41%
Number of waits	39	20

6 Conclusion and Future Research

This study focuses on the RFJSP-SDST with minimizing the makespan. A novel imitation learning and constraint programming-assisted evolutionary algorithm is proposed to solve the RFJSP-SDST problem efficiently. The ILCPEA incorporates a hybrid decoding strategy, which combines basic, matheuristic, and imitation learning-assisted methods to ensure diversity, optimal resource allocation, and a balance between computational efficiency and performance. Additionally, the algorithm enhances local search by leveraging the disjunctive graph model to identify critical paths and design four neighborhood structures that improve convergence. The introduction of a CP-based mathematical evolution operator enables a more comprehensive exploration of the solution space. Experimental results demonstrate that ILCPEA outperforms state-of-the-art methods and offers a practical solution to real workshop problems. In particular, the

proposed framework is suitable for production environments with shared and limited setup resources, such as workshops involving shared robots, human setup teams, or transportation systems where machines compete for auxiliary resources.

Despite these promising results, several limitations remain and deserve further investigation. (1) In practical deployment, challenges remain in acquiring accurate SDST data and modeling human or robotic resource availability, which may limit industrial adoption. (2) The current framework does not explicitly consider additional dynamic scheduling adjustments, such as real-time changes in machine availability or external disruptions.

Acknowledgements

This research is supported by the Funds for the National Natural Science Foundation of China [grant numbers 52205529 and 62473186], the Youth Innovation Team Program of Shandong Higher Education Institution (2023KJ206), and the Foundation of Young Talent of Lifting engineering for Science and Technology in Shandong, China (No. SDAST2024QTA074).

Author contributions

Weiyao Cheng: Conceptualization, Methodology, Writing - original draft.
Chaoyong Zhang: Supervision.
Leilei Meng: Methodology, Supervision, Funding acquisition.
Hongyan Sang: Supervision, Funding acquisition.
Biao Zhang: Supervision, Formal analysis.

Availability of data and materials

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Competing interests

The authors have no competing interests to declare that are relevant to the content of this article.

References

[1] W. Cheng, L. Meng, B. Zhang, K. Gao, and H. Sang, "Imitation Learning-Assisted Evolutionary Algorithm for Energy-Efficient Flexible Job Shop Scheduling Problem With Automated Guided Vehicles," *IEEE Trans. Evol. Comput.*, vol. 30, no. 1, pp. 171-185, 2026.

[2] L. Meng, C. Zhang, X. Shao, and Y. Ren, "MILP models for energy-aware flexible job shop scheduling problem," *J. Clean. Prod.*, vol. 210, pp. 710-723, 2019.

[3] L. Shen, S. Dauzère-Pérès, and J. S. Neufeld, "Solving the flexible job shop scheduling problem with sequence-dependent setup times," *Eur. J. Oper. Res.*, vol. 265, no. 2, pp. 503-516, 2018.

[4] D. Kress, D. Müller, and J. Nossack, "A worker constrained flexible job shop scheduling problem with sequence-dependent setup times," *OR Spectrum*, vol. 41, pp. 179-217, 2019.

[5] M. Mousakhani, "Sequence-dependent setup time flexible job shop scheduling problem to minimise total tardiness," *Int. J. Prod. Res.*, vol. 51, no. 12, pp. 3476-3487, 2013.

[6] F. M. Defersha and M. Chen, "A parallel genetic algorithm for a flexible job-shop scheduling problem with sequence dependent setups," *The international journal of advanced manufacturing technology*, vol. 49, pp. 263-279, 2010.

[7] P. Winklehner and V. A. Hauder, "Flexible job-shop scheduling with release dates, deadlines and sequence dependent setup times: A real-world case," *Procedia Comput. Sci.*, vol. 200, pp. 1654-1663, 2022.

[8] T. F. Abdelmaguid, "A neighborhood search function for flexible job shop scheduling with separable sequence-dependent setup times," *Appl. Math. Comput.*, vol. 260, pp. 188-203, 2015.

[9] S. Zhang and S. Wang, "Flexible assembly job-shop scheduling with sequence-dependent setup times and part sharing in a dynamic environment: Constraint programming model, mixed-integer programming model, and dispatching rules," *IEEE Trans. Eng. Manage.*, vol. 65, no. 3, pp. 487-504, 2018.

[10] S. Barak, S. Javanmard, and R. Moghdani, "Dual resource constrained flexible job shop scheduling with sequence-dependent setup time," *Expert Syst.*, vol. 41, no. 10, p. e13669, 2024.

[11] M. A. Boschetti, V. Maniezzo, M. Roffilli, and A. Bolufé Röhler, "Matheuristics: Optimization, simulation and control," in *International workshop on hybrid metaheuristics*: Springer, 2009, pp. 171-177.

[12] Y. Yao *et al.*, "A novel mathematical model for the flexible job-shop scheduling problem with limited automated guided vehicles," *IEEE Trans. Autom. Sci. Eng.*, 2024.

[13] X. He, Q. Pan, L. Gao, and J. S. Neufeld, "An asymmetric traveling salesman problem based matheuristic algorithm for flowshop group scheduling problem," *Eur. J. Oper. Res.*, vol. 310, no. 2, pp. 597-610, 2023.

[14] Y. Hu, L. Zhang, Q. Wang, Z. Zhang, and Q. Tang, "A matheuristic-based multi-objective evolutionary algorithm for flexible assembly jobs shop scheduling problem in cellular manufacture," *Swarm Evol. Comput.*, vol. 87, p. 101549, 2024.

[15] L. Fanjul-Peyro, F. Perea, and R. Ruiz, "Models and matheuristics for the unrelated parallel machine scheduling problem with additional resources," *Eur. J. Oper. Res.*, vol. 260, no. 2, pp. 482-493, 2017.

[16] S. Lin and K. Ying, "Optimization of makespan for no-wait flowshop scheduling problems using efficient matheuristics," *Omega-Int. J. Manage. Sci.*, vol. 64, pp. 115-125, 2016.

[17] J. Fan, C. Zhang, S. Tian, W. Shen, and L. Gao, "Flexible job-shop scheduling problem with variable lot-sizing: An early release policy-based matheuristic," *Comput. Ind. Eng.*, p. 110290, 2024.

[18] J. Fan, C. Zhang, W. Shen, and L. Gao, "A matheuristic for flexible job shop scheduling problem with lot-streaming and machine reconfigurations," *Int. J. Prod. Res.*, vol. 61, no. 19, pp. 6565-6588, 2023.

[19] Y. Hu, L. Zhang, Z. Zhang, Z. Li, and Q. Tang, "Matheuristic and learning-oriented multi-objective artificial bee colony algorithm for energy-aware flexible assembly job shop scheduling problem," *Eng. Appl. Artif. Intell.*, vol. 133, p. 108634, 2024.

[20] Y. Hu, L. Zhang, Z. Zhang, Z. Li, and Q. Tang, "Flexible assembly job shop scheduling problem considering reconfigurable machine: A cooperative co-evolutionary matheuristic algorithm," *Appl. Soft Comput.*, vol. 166, p. 112148, 2024.

[21] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1-35, 2017.

[22] H. Ingimundardottir and T. P. Runarsson, "Discovering dispatching rules from data using imitation learning: A case

study for the job-shop problem," *J. Sched.*, vol. 21, pp. 413-428, 2018.
[23] L. Li *et al.*, "Learning to optimize permutation flow shop scheduling via graph-based imitation learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024, pp. 20185-20193.
[24] J. Lee and H. Kim, "Graph-Based Imitation Learning for Real-Time Job Shop Dispatcher," *IEEE Trans. Autom. Sci. Eng.*, 2024.
[25] J. Li, J. Li, K. Gao, and P. Duan, "A hybrid graph-based imitation learning method for a realistic distributed hybrid flow shop with family setup time," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2024.
[26] I. Echeverria, M. Murua, and R. Santana, "Leveraging constraint programming in a deep learning approach for dynamically solving the flexible job-shop scheduling problem," *Expert Syst. Appl.*, vol. 265, p. 125895, 2025.
[27] J. C. Beck, T. K. Feng, and J. Watson, "Combining constraint programming and local search for job-shop scheduling," *INFORMS J. Comput.*, vol. 23, no. 1, pp. 1-14, 2011.
[28] V. Heinz, A. Novák, M. Vlk, and Z. Hanzálek, "Constraint programming and constructive heuristics for parallel machine scheduling with sequence-dependent setups and common servers," *Comput. Ind. Eng.*, vol. 172, p. 108586, 2022.
[29] L. Meng, W. Cheng, B. Zhang, W. Zou, and P. Duan, "A novel hybrid algorithm of genetic algorithm, variable neighborhood search and constraint programming for distributed flexible job shop scheduling problem," *Int. J. Ind. Eng. Comput.*, vol. 15, no. 3, pp. 813-832, 2024.
[30] G. A. Kasapidis, D. C. Paraskevopoulos, I. Mourtos, and P. P. Repoussis, "A unified solution framework for flexible job shop scheduling problems with multiple resource constraints," *Eur. J. Oper. Res.*, vol. 320, no. 3, pp. 479-495, 2025.
[31] L. Meng *et al.*, "Novel MILP and CP models for distributed hybrid flowshop scheduling problem with sequence-dependent setup times," *Swarm Evol. Comput.*, vol. 71, p. 101058, 2022.
[32] C. Luo, X. Li, W. Gong, and L. Gao, "Affinity Propagation Hierarchical Memetic Algorithm for Multimodal Multi-Objective Flexible Job Shop Scheduling With Variable Speed," *IEEE Trans. Evol. Comput.*, 2025.
[33] R. Li, L. Wang, W. Gong, and F. Ming, "An evolutionary multitasking memetic algorithm for multi-objective distributed heterogeneous welding flow shop scheduling," *IEEE Trans. Evol. Comput.*, 2024.
[34] X. Han *et al.*, "A dual population collaborative genetic algorithm for solving flexible job shop scheduling problem with AGV," *Swarm Evol. Comput.*, vol. 86, p. 101538, 2024.
[35] F. Zhao, F. Yin, L. Wang, and Y. Yu, "A Co-Evolution Algorithm With Dueling Reinforcement Learning Mechanism for the Energy-Aware Distributed Heterogeneous Flexible Flow-Shop Scheduling Problem," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2024.
[36] R. Zhang, H. Yu, K. Gao, Y. Fu, and J. H. Kim, "A Q-learning based artificial bee colony algorithm for solving surgery scheduling problems with setup time," *Swarm Evol. Comput.*, vol. 90, p. 101686, 2024.
[37] L. Meng *et al.*, "An improved genetic algorithm for solving the multi-AGV flexible job shop scheduling problem," *Sensors*, vol. 23, no. 8, p. 3815, 2023.
[38] L. Zhao *et al.*, "MILP modeling and optimization of flexible job shop scheduling problem with preventive maintenance," *Comput. Ind. Eng.*, vol. 201, p. 110861, 2025.
[40] Li H, Gao L, Fan Q, et al. An end-to-end decentralised scheduling framework based on deep reinforcement learning for dynamic distributed heterogeneous flowshop scheduling[J]. *Int. J. Prod. Res*, 2025, 63(12): 4368-4388.
[41] Yang Z, Li X, Gao L, et al. A novel topological neighborhood structure for flexible job shop scheduling problem with variable sublots[J]. *Comput. Oper. Res.*, 2025, 182: 107120.
[42] Cheng W, Zhang C, Meng L, et al. Collaborative multi-CP model and meta-feedback learning-assisted matheuristic for solving the flexible job shop scheduling problem with sequence-dependent setup times[J]. *Swarm Evol. Comput.*, 2025, 99: 102173.

Weiyao Cheng received the B.S. degree from the School of Computer Science, Liaocheng University, Liaocheng, China, He is currently pursuing the Ph.D. degree in mechanical engineering with the Huazhong University of Science and Technology, Wuhan, China. He has authored 10 refereed articles. His research mainly focuses on modeling and optimization of scheduling problems.

Chaoyong Zhang received the M.S. degree in mechatronic engineering from the University of Science and Technology Beijing, Beijing, China, in 1999, and the Ph.D. degree in mechatronic engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2007. He is currently a Professor with the School of Mechanical Science and Engineering, HUST. He has authored two books and over 50 SCI articles as the first author/corresponding author. His research interests include modeling and optimization control for green manufacturing, remanufacturing systems, sustainable manufacturing, including clean and high efficient manufacturing processes, and the power consumption model of machine tools.

Leilei Meng received the B.S. degree in mechanical engineering from Chang'an University, Xi'an, China, in 2014, and the Ph.D. degree from the School of Mechanical Science and Engineering, Huazhong University of Science and Technology (HUST), China, in 2020. He is currently an associate professor with the School of Computer Science, Liaocheng University. He has authored more than 140 refereed articles. His research mainly focuses on modeling, optimization of scheduling problems, tool wear prediction, and sustainable manufacturing.

Hongyan Song received the M.S. degree in industrial engineering from the School of Computer Science, Liaocheng University, Liaocheng, China, in 2010, and the Ph.D. degree in industrial engineering from the Huazhong University of Science Technology, Wuhan, China, in 2013. Since 2003, she has been with the School of Computer Science, Liaocheng University, where she became a professor in 2021. She has authored more than 150 refereed papers. Her current research interests include intelligent optimization and scheduling.

Biao Zhang received the B.S. degree from the School of Computer Science and Technology, Shandong University of Technology, Zibo, China, the M.S. degree from the School of Computer Science, Liaocheng University, Liaocheng, China, and the Ph.D. degree from the School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan, China, in 2012, 2015, and 2019, respectively. He is currently an associate professor with the School of Computer Science, Liaocheng University. His current research interests include discrete optimization and scheduling.