An Improved Sparrow Search Algorithm for Flexible Job-Shop Scheduling Problem with Setup and Transportation Time

Yi Li[1], Song Han[2], Zhaohui Li[3], Fan Yang[4], Zhengyi Sun[5]
School of Maritime Economics and Management, Dalian Maritime University, Dalian China[1, 2, 3]
Hangzhou Hollysys Automation Co., Ltd. Xi 'an branch, Xi'an China[4]
Graduate School of Information, Production and Systems, Waseda University, Kitakyushu Japan[5]

***Abstract*—This study addresses the low production efficiency in manufacturing enterprises caused by the diversification of order products, small batches, and frequent production changeovers. Focusing on minimizing the makespan, this study establishes a Flexible Job-Shop Scheduling Problem (FJSP) model incorporating machine setup and workpiece transportation times, and proposes an improved sparrow search algorithm to effectively solve the problem. Based on the sparrow search algorithm, this study proposes a novel location update strategy that expands the search direction in each dimension and strengthens each individual's local search capability. In addition, a critical-path-based neighborhood search strategy is introduced to enhance individual search efficiency, and an earliest completion time priority rule is employed during population initialization to further improve solution quality. Several experiments are conducted to validate the effectiveness of the improved strategy, and the results are compared with those obtained using the particle swarm optimization and gray wolf optimization algorithms to demonstrate the efficiency of the proposed model and algorithm. The improved sparrow search algorithm can effectively generate feasible solutions for large-scale problems, provide practical manufacturing scheduling schemes, and enhance the production efficiency of manufacturing enterprises.**

Keywords*—Flexible job shop scheduling; machine setup; transportation; sparrow search algorithm; earliest completion time priority

I. INTRODUCTION

The Flexible Job Shop Scheduling Problem (FJSP) is a key area in modern manufacturing. The growing complexity of market demands, such as product diversification, small-batch production, and frequent changeovers, has intensified the need to consider setup and transportation times in production scheduling. This makes FJSP research incorporating these factors, a critical academic focus.

Scholars have conducted extensive research on the flexible job shop scheduling problem under single constraints, either setup time or transportation time. Defersha et al.[1], investigated the flexible job shop scheduling problem considering worker-machine setup times and proposed an improved simulated annealing algorithm to solve it. Li et al.[2], proposed an improved artificial immune system algorithm to solve the flexible job shop scheduling problem considering setup scenarios. Peng et al.[3], investigated the multi-objective flexible job shop scheduling problem with job transportation time and learning effect constraints, and proposed a hybrid discrete multi-objective imperialist competitive algorithm to solve the model. Zhang Guohui et al. [4], examined the impact of machine installation, positioning, and other adjustment times on the flexible job shop scheduling problem, and proposed an improved genetic algorithm to solve the problem. Sadrzadeh [5], proposed a hybrid artificial immune-particle swarm optimization algorithm and validated its effectiveness through numerical experiments. Zhang et al.[6], designed a genetic algorithm with a tabu search procedure to solve the flexible job shop scheduling problem with transportation constraints and limited processing times. The aforementioned scholars have proposed various algorithms to address the FJSP with either setup or transportation times separately considered. However, these studies have overlooked the interactions among processing, setup, and transportation times. Setup times affect the start time of processing tasks, whereas processing times determine the start time of transportation tasks. The combined effects of setup and transportation times result in varying machine waiting times. Therefore, flexible job shop scheduling problem that simultaneously incorporates setup and transportation times is more consistent with real-world production scenarios.

For the flexible job shop scheduling problem that incorporates both setup and transportation times, An et al.[7], proposed a hybrid multi-objective evolutionary algorithm based on a Pareto elite storage strategy, aiming at minimizing the makespan, total delay, total production cost, and total energy consumption. Li et al.[8], simultaneously optimized energy consumption and makespan, employing an improved Jaya algorithm to solve the problem. Zhang et al.[9], proposed an effective heuristic algorithm to minimize the makespan and total energy consumption. Sun et al.[10], developed a hybrid multi-objective evolutionary algorithm aimed at minimizing makespan, total workload, critical machine workload, and penalties for early or late completion. Rossi [11], investigated the flexible job shop scheduling problem incorporating both transportation and setup times, employing an ant colony algorithm enhanced with pheromone mechanisms. In summary, the primary approaches for solving the flexible job shop scheduling problem encompass exact algorithms based on mathematical programming, as well as intelligent evolutionary methods such as the Genetic Algorithm (GA) [12], Tabu Search

(TS) [13], Ant Colony Optimization (ACO) [14], and Particle Swarm Optimization (PSO) [15]. Traditional algorithms such as genetic algorithms and tabu search often encounter limitations when addressing these problems, including high dimensionality, slow convergence, and challenging parameter tuning. In 2020, Xue et al., proposed the Sparrow Search Algorithm (SSA) [16], a novel population-based intelligent optimization method characterized by simple principles, few parameters, and ease of implementation, and has been widely applied in various fields [17]. Although the SSA algorithm has also been applied to solve the FJSP, its application to the FJSP with simultaneous consideration of setup and transportation times remains relatively rare.

Based on the aforementioned background, this study incorporates the real production scenario of Dalian BL Technology Co., Ltd. and formulates an integer programming model for the flexible job shop scheduling problem, which considers both setup and transportation times, aiming to minimize the makespan. The effectiveness of the proposed algorithm is verified using the CPLEX solver. As the data scale increases, it becomes difficult for exact algorithms to solve the problem in a short time. This study introduces enhanced strategies, culminating in the design of an Improved Sparrow Search Algorithm (ISSA) to solve the problem. Finally, the effectiveness of these enhanced strategies and the efficiency of the ISSA algorithm are validated using the small-scale Kacem and medium-to-large-scale Brandimarte benchmark instances.

The remainder of this paper is organized as follows: Section II formulates the problem and constructs the mathematical model. Section III presents the encoding scheme and proposes the improved sparrow search algorithm. Computational experiments are conducted in Section IV, followed by results and discussions in Section V. Finally, conclusions are provided in Section VI.

II. Problem Description and Model Construction

A. Problem Description

Dalian BL Technology Co., Ltd. is a multi-sector, order-driven manufacturing enterprise. The orders they receive typically consist of small batches and a wide variety of parts. When processing different types of parts, the machines require adjustments such as changing tool heads and adjusting machine parameters. Additionally, when metal parts proceed to the next processing step, they often need to be transferred to different machines, and manual transport equipment is employed to move the parts. Building on this manufacturing scenario, the workshop scheduling problem can be formulated as a flexible job shop scheduling problem (FJSP) that incorporates both setup and transportation times, described as follows: there is a set of n jobs, denoted by $J = \{J_1, J_2, \ldots, J_n\}$, and a set of m machines, denoted by $M = \{M_1, M_2, \ldots, M_m\}$. Each job J_i consists of j operations, with the j -th operation of job J_i represented by O_{ij}. Each operation can be processed on one or more machines; however, each machine can process only one operation at a time. Once an operation starts on a machine, it cannot be interrupted. Operations within the same job must adhere to a prescribed sequence, whereas there are no sequencing constraints among operations from different jobs. At any given time, each job can be processed on only one machine. Before any machine can process a job, it must be adjusted by workers according to that job's characteristics; moreover, the machine requires re-adjustment when switching between jobs. When transferring a job's operation to a different machine, transport equipment is required to move the job.

The problem follows the standard constraints of the flexible job shop scheduling problem while additionally accounting for the effects of machine setup and transportation on the scheduling process. Based on real-world conditions and the scope of this research, the following constraints and assumptions are proposed:

- If a job is processed consecutively on the same machine, no transportation is required.
- If two or more consecutive operations on a machine belong to the same job, no setup time is required for the subsequent operation.
- Loading and unloading times are included in the overall transportation time.
- Human resources and transport equipment are sufficiently available and can respond in real-time.

B. Scenario Analysis

In the actual manufacturing scenario, processing can begin only after setup is completed, thereby influencing the processing start time. Similarly, transportation can commence only once an operation finishes, affecting the start time of transportation. Furthermore, subsequent processing can begin only after the setup has been completed and the job has been transferred to the next machine. The combined effects of setup and transportation times influence the machine's waiting time. Consequently, setup, transportation, and processing times are interrelated.

Taking the extended Kacem 4×5 dataset as an example, if scheduling is carried out without accounting for setup and transportation times, the resulting plan is shown in Fig. 1(a). If this scheduling plan is applied directly in the workshop, a significant delay in the overall makespan will result, as illustrated in Fig. 1(b). However, after incorporating the effects of setup and transportation times on the makespan, the proposed model optimizes the scheduling plan, and the final outcome, depicted in Fig. 1(c), achieves a shorter makespan compared to the previous plan.

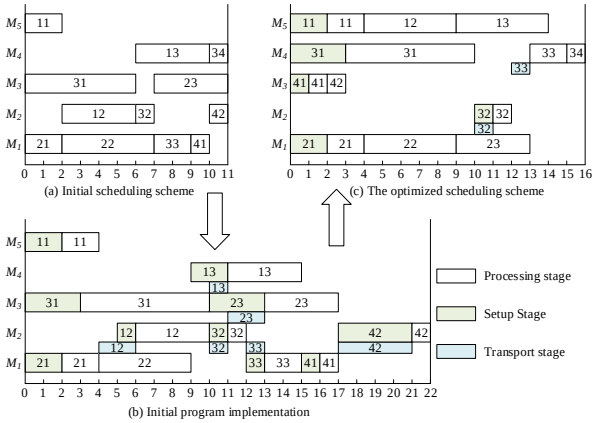


Fig. 1. Comparison of scheduling schemes

C. Model Construction

The definitions of the model parameters are provided in Table I.

TABLE I. PARAMETERS OF FJSP MODEL WITH SETUP TIME AND TRANSPORTATION TIME

Parameter	Definition
J	Job set
M	Machine set
i, p	Job index
J_i	A set of operations for job i
j, q	Operation index
k, l	Machine index
C_i	Completion time of job i
O_{ij}	Operation j of job i
v_{ij}	Start transportation time of O_{ij}
u_{ij}	End transportation time of O_{ij}
s_{ij}	Start setup time of O_{ij}
$Cmax$	Maximum completion time
e_{ij}	Ends setup time of O_{ij}
g_{ij}	Processing starts time of O_{ij}
h_{ij}	End processing time of O_{ij}
T_{ijk}	Processing time on machine k of O_{ij}
P_{lk}	Transportation time from machine l to k
W_{ijk}	Setup time on machine of O_{ij}
z_{ij}	For O_{ij} 1 indicates that setup is required, 0 indicates that setup is not required
x_{ijk}	For O_{ij} 1 indicates that processing on machine k, 0 indicates not
y_{ijkpq}	ForO_{ij} 1 indicates that processing before O_{pq} on machine k, 0 indicates not
L	A large positive number
a	Virtual workpieces serve as start and end markers on the machine, helping to enforce tight-front and tight back constraints

Based on the problem description, the model is constructed as follows:

$$mainCmax = maxC_i \tag{1}$$

$$h_{ij} \le v_{ij+1} \tag{2}$$

$$\sum_{k\in M} x_{ijk} = 1 \tag{3}$$

$$z_{ij} = \begin{cases} 1 \\ 1 - \sum_{k\in M}\sum_{p\in\{J_1,\dots,J_{j-1}\}} y_{ipkij} \end{cases} \tag{4}$$

$$h_{ij} = g_{ij} + \sum_{k\in M}\left(T_{ijk}\cdot x_{ijk}\right) \tag{5}$$

$$e_{ij} = s_{ij} + \sum_{k\in M}\left(W_{ijk}\cdot x_{ijk}\right)\cdot z_{ij} \tag{6}$$

$$u_{ij} = v_{ij} + \sum_{j,k\in M}\left(P_{ijk}\cdot x_{i(j-1)k}\right)\cdot x_{ijk} \tag{7}$$

$$h_{ij} \le s_{ij} + L\cdot\left(\sum_{k\in M} y_{ijkpq}\right) \tag{8}$$

$$\sum_{p\in J\cup\{a\},q\in J_p} y_{ijkpq} = x_{ijk} \tag{9}$$

$$\sum_{p\in J\cup\{a\},q\in J_p} y_{pqkij} = x_{ijk} \tag{10}$$

$$s_{ij} \le e_{ij} \le g_{ij} \le h_{ij} \le C_i \tag{11}$$

$$v_{ij} \le u_{ij} \le g_{ij} \le h_{ij} \le C_i \tag{12}$$

Eq. (2) stipulates those operations of the same job must be processed in sequence, and that transportation can commence only after the preceding operation is finished. Eq. (3) indicates that each process must be assigned to exactly one machine for processing. Eq. (4) indicates whether an operation requires setup. Eq. (5) represents the processing time constraints for the job. Eq. (6) represents the setup phase time constraints for the job. Eq. (7) represents the transportation phase time constraints for the job. Eq. (8) imposes timing constraints between adjacent operations and ensures that only one operation (whether setup or processing) can be performed on a machine at a time. Eq. (9) and Eq. (10) stipulate that if a job is being processed on a machine, there must be one preceding and one succeeding operation (including virtual operations). Eq. (11) and Eq. (12) represents the time constraints for each phase of the operation, requiring that the job must arrive at the machine and complete the setup before production begins. Additionally, the setup and transportation operations can occur independently.

III. ALGORITHM DESIGN

The flexible job shop scheduling problem that considers both setup and transportation times, as investigated in this study, is an NP-hard problem. As the problem size grows, exact algorithms struggle to produce solutions within a short time. Considering the sparrow search algorithm's advantages—few parameters and ease of implementation—this study adopts and refines it to efficiently solve the aforementioned mixed-integer programming model.

A. Encoding and Decoding

1) In Flexible job shop scheduling research, encoding primarily addresses two aspects: operation sequencing and machine selection. To address this challenge, a two-stage encoding scheme—operation sequence and machine sequence—is designed, as illustrated in Fig. 2.

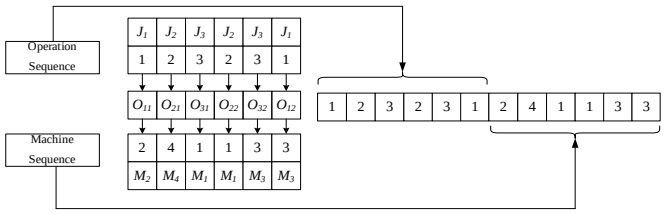


Fig. 2. An example of encoding.

2) Operation Sequence (OS): Each element in the sequence represents an operation for a job, and its position in the encoded sequence determines the order in which operations are performed. For example, if the OS sequence is1-2-1-3-2-3-1, it means the sequence of operations for these three jobs is $O_{11} - O_{21} - O_{12} - O_{31} - O_{22} - O_{32} - O_{13}$ .This encoding method

ensures the sequential constraints among multiple operations of each workpiece.

3) Machine Sequence (MS): Each element in the sequence represents a machine, specifying which machine processes the corresponding operation in the OS sequence. For example, if the MS sequence is 1-3-1-2-3-1-3, then O_{11} is processed on machine M_1, and operation O_{21} is processed on machine M_3.

By applying the Ranked Order Value (ROV) rule, one can map continuous individual vectors to discrete individual vectors. This process consists of two parts: encoding conversion for operation sequencing and encoding conversion for machine assignment. After decoding, one must evaluate the resulting machining scheme's quality and determine whether forward insertion of the workpiece is necessary.

- The operation sequence conversion steps are illustrated in Fig. 3.
- The machine encoding is mapped according to Eq. (13).

$$m_o = round\left(\frac{(\lambda+m)(m-1)}{2m} + 1\right) \quad (13)$$

The $o \in [1, d]$ represents the ordinal of the operation sequence, where d is the total number of operations. The function, $round()$ performs rounding. The parameter $\lambda \in [-m, m]$ indicates that the coded position corresponds to the individual's location in continuous space. The variable m denotes the total number of machines, and m_o denotes the machine number selected for the corresponding operation O_{ij}.

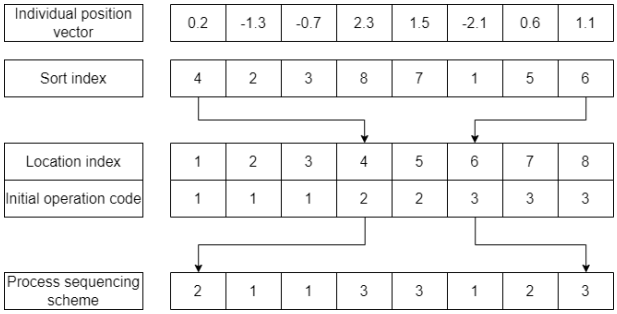


Fig. 3. ROV Mapping.

B. Discoverers Location Update Strategy Optimization

An analysis of the discoverers' location update strategy in the sparrow population shows that when R_2 is lower than ST, the coefficient's range gradually shrinks from the initial [0,1] interval to roughly [0,1] as the number of iterations i increases [16]. In particular, when i is small, the coefficient is more l likely to be close to 1, thereby reducing the sparrow's range of activity in each dimension of the search space. Because finders constitute only a small fraction of the entire population, i remains relatively small, causing the positional update factor to tend toward 1. To address this issue, if R_2 is less than ST, Eq. (14) can be used for the position update; otherwise, Eq. (15) is adopted.

$$X_{ij}^{t+1} = X_{ij}^{t} \cdot \left(2t + (-1)^t \cdot exp\left(\frac{-i}{a \cdot iter}\right)\right) \quad (14)$$

$$X_{ij}^{t+1} = X_{ij}^{t} + Q \cdot L \quad (15)$$

C. Critical Path-Based Neighborhood Search

To further enhance the SSA algorithm's performance, a critical-path-based neighborhood search method is integrated into the basic SSA framework.

Processes located on the critical path often play a decisive role in determining the final quality of the overall scheduling scheme, as their completion times directly dictate the length of the entire production cycle. By adjusting these critical processes along with their adjacent operations, the algorithm explores additional solution spaces, thereby enhancing the potential to discover superior solutions. The procedure is as follows:

- Identify critical and non-critical operations;
- Randomly select one operation from the critical-operation set and one from the non-critical-operation set for swapping;
- New sequence feasibility check, the operation may be exchanged to the machine without processing capacity. The machine selection is carried out through the Earliest Completion Time First rule;
- Fitness calculation- Assign machines based on the workpiece coding sequence and machine coding, and perform forward insertion strategy to explore better results;
- Population update.

The pseudocode is as follows:

Algorithm 1: Critical path identification

```
Input: Operation Set
Begin:
    initialize CO
    for each operation O_ij:
        if h_ij = makespan then:
            put O_ij in CO
        end if
    end for
    while CO != null then:
        delete first operation O' of CO
        T = h_ij of O'
        for O_ij in Operation Set
            if h_ij = T then:
                put O_ij in CO
            end if
        end for
    end while
    return all marked operation
End
```

D. Population Initialization

In the sparrow optimization algorithm, constructing the initial population is the first step, influencing the subsequent optimization process and outcomes. Although random initialization maintains the abundance and diversity of the population, the quality of individuals remains inconsistent. In

the initial phase of the algorithm, it may be difficult to quickly find a high-quality solution, requiring numerous iterations to gradually approach the optimal solution. This process raises the algorithm's computational cost and execution time. To enhance the algorithm's performance, this study employs random generation and ECT rule-based initialization to produce 50% of the population. The ECT rule dynamically calculates the completion time of each operation on every machine and assigns tasks to the machine with the earliest completion, thereby eliminating unreasonable machine selections during initialization and producing an initial solution that is both high in quality and diverse.

The pseudocode is as follows:

Algorithm 2: ECT rule

Input: Current operation (O_{ij}), process time on each machine (T_{ijk}), setup time on each machine (W_{ijk}), transportation time between M_l (processO_{ij-1}) to current machine M_k(P_{lk})

Begin:
- **initialize** $t, p, w, completion\_time, et$
- **for** each machineM_k**:**
  - **If** $M_k$can process O_{ij}**then:**
    - $p = T_{ijk}, w = W_{ijk}$
    - **If** O_{ij} is the first operation of the Job **or** the first operation on M_k **or** O_{ij} and O_{pq} are from same job **then:**
      - $st = 0$
    - **end if**
    - **if** O_{ij} is the first operation of the Job or O_{ij-1} is processed on M_k **then:**
      - $p = 0$
    - **else**
      - $p = P_{lk}$
    - **end if**
    - completion_time = $max(h_{ij-1} + tt, h_{pq} + w) + t$
    - **if** completion_time ≤ et **then:**
      - mark current machine M_k and update et
    - **end if**
  - **end if**
- **end for**
- **return** marked machine

End

IV. Analysis of Numerical Experiments

This study performs an ablation experiment to validate both the effectiveness of the proposed algorithmic improvement strategy and the algorithm's overall efficiency. Additionally, small-scale and medium-to-large-scale experiments were conducted to further assess the algorithm's efficiency. The experiments were implemented in Java, running on an Intel(R) Core(TM) i5-7500 CPU @ 3.40GHz processor with 8GB RAM and the Windows 10 Professional operating system.

The experimental dataset consists of 15 newly created instances (MK01–MK10, Kacem01–Kacem05), which is generated by the small-scale Kacem dataset and the medium-scale Brandimarte dataset [18]. The commissioning time and shipping time are generated according to relevant strategies [19].

For the algorithm parameters, the population size exhibits a critical trade-off in optimization algorithms: an undersized population is prone to premature convergence to local optima, whereas an excessively large population imposes prohibitive computational overhead. Insufficient iteration cycles compromise convergence completeness while introducing substantial computational redundancy. The proportional allocation between discoverers and followers critically modulates the equilibrium between global exploration and local exploitation capacities within the algorithm framework. Through systematic orthogonal experimental design, the optimal parameter configuration was determined as follows: The population size was set to $N = 100$, the maximum number of iterations to $N\_iter = 400$, the discoverer proportion to $PD = 20\%$, and the vigilant proportion to $SD = 80\%$.

A. Validation of the Improvement Strategy's Effectiveness

The Improved Sparrow Search Algorithm integrates three strategies into the standard algorithm: SSA1 denotes an optimized discoverer location update strategy, SSA2 represents population initialization via an ECT heuristic, and SSA3 adopts a critical-path-based neighborhood search strategy. This study designed eight sets of experiments to compare the strategies presented in TABLE II. The eight algorithms were each run independently ten times on the Brandimarte dataset, recording their best, average, and variance values, as well as the solution time.

TABLE II. Algorithm Comparison Strategy

	SSA	SSA1	SSA2	SSA3	SSA12	SSA13	SSA23	SSA123
Strategy 1	○	●	○	○	●	●	○	●
Strategy 2	○	○	●	○	●	○	●	●
Strategy 3	○	○	○	●	○	●	●	●

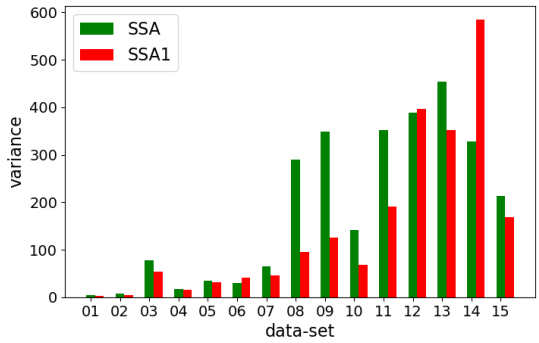
Note: ● indicates that the policy is applied; ○ indicates that the policy is not applied

1) Analysis of the discoverer's location update strategy: Strategy I modifies the parameters of the sparrow population's position update formula, increasing the step size and direction of the position update and enhancing the global search capability of the sparrow search algorithm to avoid local optima. In independent runs, the tenth solution comes closer to the optimum, showing reduced variance. As illustrated in Fig. 4(a), SSA1 outperforms SSA in variance across 12 of the 15 datasets, exhibiting a reduction of over 50% in Mk08 and Mk09. Fig. 4(b) to (d) presents comparative trials of the other groups incorporating Strategies II and III.

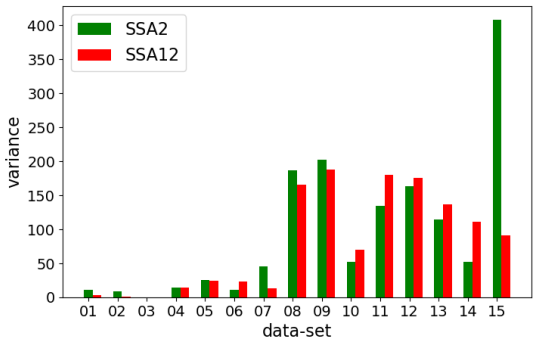
2) Analysis of ECT rule strategies: Among the four algorithms listed in Table II—SSA2, SSA12, SSA23, and SSA123—incorporate the finder location update strategy. To assess the impact of Strategy II, it is removed from these algorithms and compared with SSA, SSA1, SSA3, and SSA13. As a specific strategy, the ECT rule is tailored to the characteristics of this problem and is thus highly suited to the research in this chapter. The ECT rule quickly locates machines with shorter completion times, complementing the sparrow search algorithm. In conjunction with its global search

capability, this improves solution quality and efficiency. As shown in Fig. 5(a) to (d), when the effects of Strategies I and III are excluded, incorporating improved Strategy II yields superior solution quality. Since the sparrow algorithm relies on the current optimal solution when the sparrow population undergoes positional updating during the iterative process, high quality initial values can improve the solution quality. Results from ten independent experimental runs reveal a notable decrease in both the best and average values across all datasets, with a more pronounced effect on larger datasets. The quality of the initial population solution generated by the ECT rule and the random generation method is shown in Fig. 6(a), and the variance of the solution is shown in Fig. 6(b).

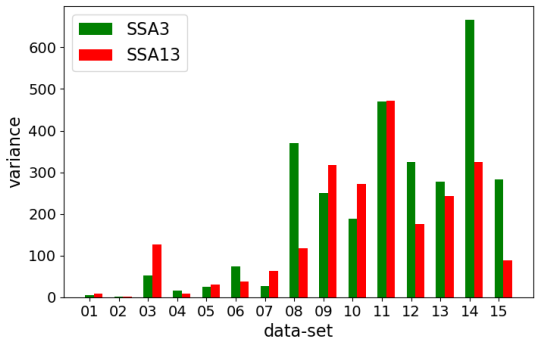
3) Critical path-based neighborhood search strategy analysis: Among the algorithms listed in Table II, SSA3, SSA13, SSA23, and SSA123 apply the location update strategy and culling Strategy II is compared with algorithms SSA, SSA1, SSA2, and SSA12. Incorporating Strategy III increases the solution time while reducing the average solution value. Since the processes on the critical path dictate the final completion time, each iteration later applies Strategy III to swap critical and non-critical processes. While this two-step operation of identifying and exchanging key processes increases computation time, it also enables a stronger local search capability. Fig. 7. (a) to (d) compares the effects of applying exclusion Strategies I and II and improvement strategy III on the algorithm's convergence performance. Incorporating Strategy III, clearly enhances the sparrow population's capacity for precise searching, leading to higher-quality solutions across iterations. The algorithm augmented by the improved Strategy III achieves an even better optimal solution.



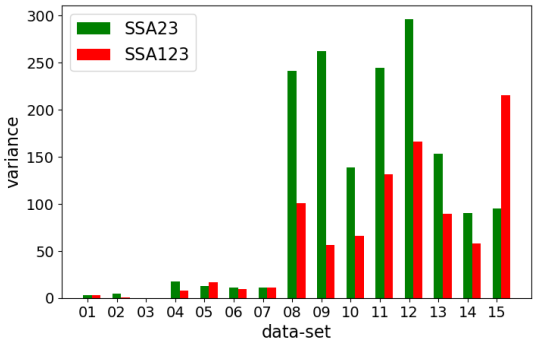
(a) Comparison of the variance of SSA and SSA1



(b) Comparison of the variance of SSA2 and SSA12

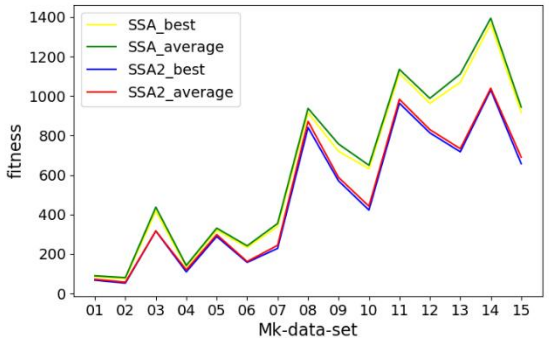


(c) Comparison of the variance of SSA3 and SSA13

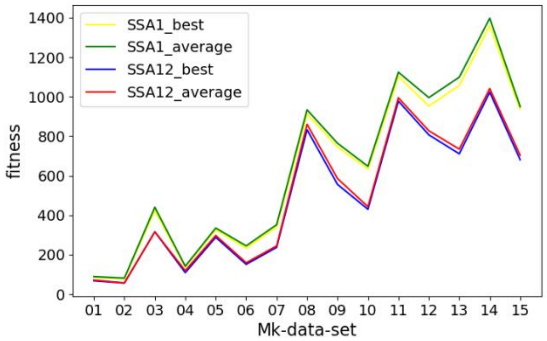


(d) Comparison of variance of SSA23 and SSA123

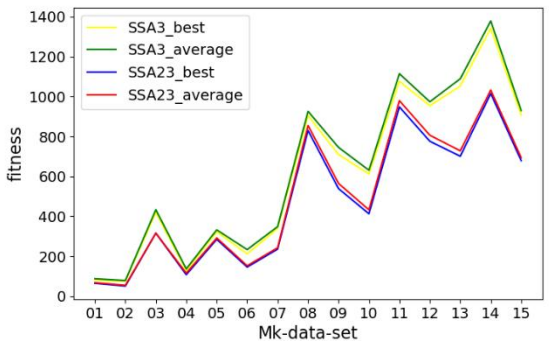
Fig. 4. Comparison of variance between algorithms with Strategy I.



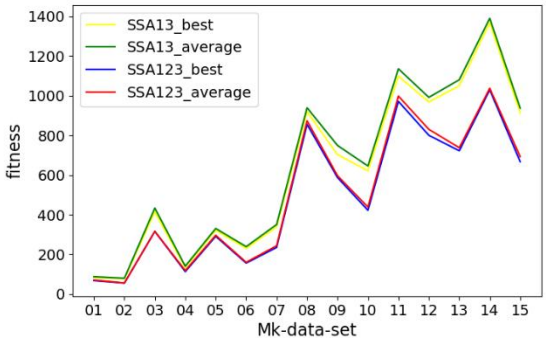
(a) Comparison of SSA and SSA2 results



(b) Comparison of SSA1 and SSA12 results

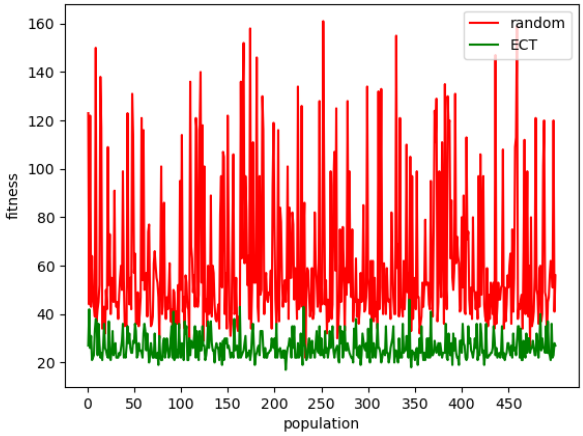


(c) Comparison of SSA3 and SSA23 results

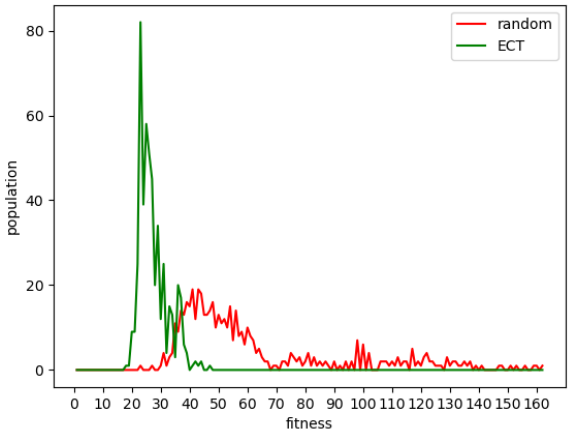


(d) Comparison of SSA13 and SSA123 results

Fig. 5. Comparison of fitness between algorithms with Strategy II.

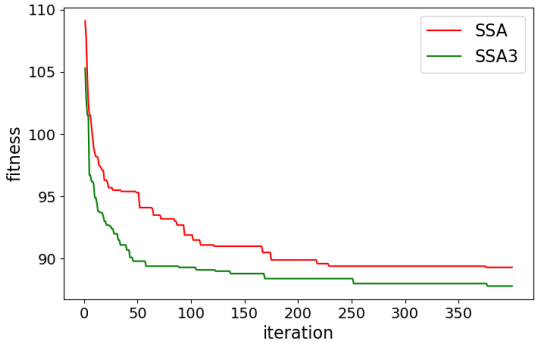


(a) Individual fitness of populations

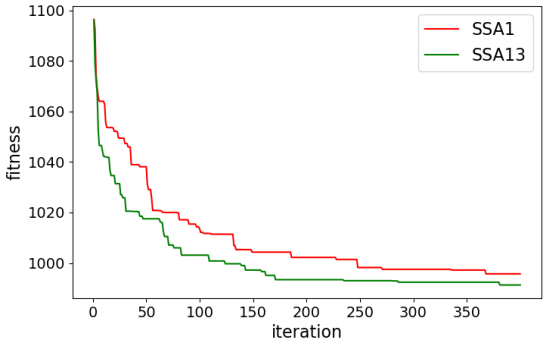


(b) Statistics on the fitness of the populations

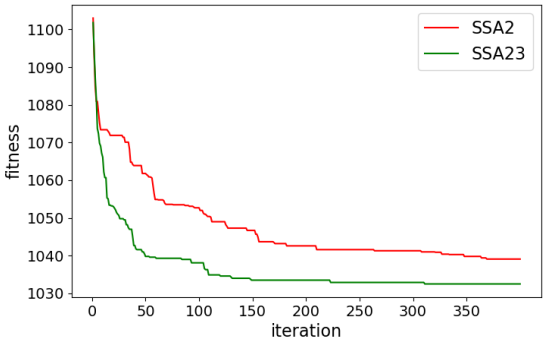
Fig. 6. Comparison of iterative process between algorithms with Strategy III.



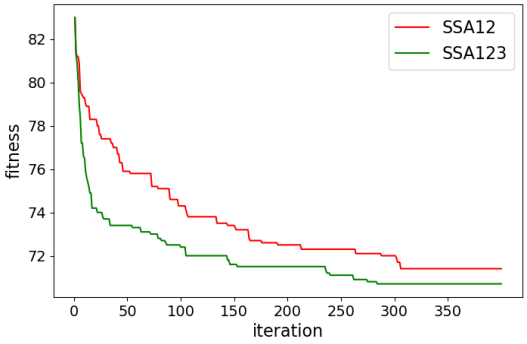
(a) SSA and SSA3 iterative process on Mk01



(b) SSA1 and SSA13 iterative process on Mk12



(c) SSA2 and SSA23 iterative process on Mk14



(d) SSA12 and SSA123 iterative process on Mk01

Fig. 7. Convergence under Strategies I-III; Strategies III shows improved precision and solution quality.

B. Small-Scale Experimental Validation Analysis

In the small-scale experiments, Java was used to invoke the CPLEX solver for comparison with the Improved Sparrow Search Algorithm. The solution results for the small-scale Kacem benchmark instances are presented in TABLE III. The Improved Sparrow Search Algorithm's solution time grows slowly as the problem size increases. Moreover, the difference between its objective function value and that of the exact solution via CPLEX remains small—specifically, the gap between the optimal solutions is only 1. However, from the kacem8-8 instance onwards, ISSA runs significantly faster than the solver, demonstrating the effectiveness of the Improved Sparrow Search Algorithm.

TABLE III. COMPARISON OF CPLEX AND ISSA ON KACEM DATA SET

Dataset	CPLEX		ISSA	
	f	t/s	f	t/s
kacem4-5	16	0.22	17	3.63
kacem8-8	22	32.03	24	10.61
kacem10-7	17	322.98	18	9.97
kacem10-10	11	37.13	12	11.11
kacem15-10	22	115200	23	35.59

C. Analysis of Large-Scale Experimental Validation

TABLE IV. presents the experimental results of the Improved Sparrow Search Algorithm compared with the standard sparrow search algorithm, the standard gray wolf optimization algorithm, and the genetic algorithm. The experiment uses the Brandimarte dataset, running each set of algorithms independently ten times. The average value was taken as the solution for each algorithm, and the performance gap between the three algorithms and the best solution among them was also recorded. From Table IV, it can be observed that the Improved Sparrow Search Algorithm shows a significant improvement in all 15 instances, with a minimum improvement rate of 7.96% and an average improvement rate of 26.48%. Moreover, the optimal values obtained by ISSA are consistently better than those achieved by the gray wolf algorithm and the genetic algorithm. The enhanced position updating strategy strengthens the sparrow search algorithm's global search capability, helping it avoid local optima and consistently discover superior solutions across all the 15 algorithms. In all 15 test instances, the Improved Sparrow Search Algorithm achieves the optimal solution. G_g denotes the gap between the optimal values of the GWO, GA, and ISSA solutions. The GWO algorithm's smallest gap is 9.0%, with an average gap of 33%. Meanwhile, when comparing the GA algorithm to ISSA, the smallest gap is 20.3%, and the average gap is 56%. These findings confirm that the ISSA algorithm developed in this study exhibits superior stability, convergence, and efficiency when solving FJSP problems involving setup and transportation times.

TABLE IV. COMPARISON OF ISSA, SSA, GWO AND GA

Dataset	ISSA	SSA		GWO		GA	
	f	f	G_g	f	G_g	f	G_g
Mk01	70.7	84	18.81%	91.0	28.7%	92.3	30.6%
Mk02	56.0	75	33.93%	83.1	48.4%	77.1	37.7%
Mk03	316.0	416	31.65%	439.4	39.1%	543.7	72.1%
Mk04	118.2	135	14.21%	141.2	19.5%	154.9	31.0%
Mk05	296.4	320	7.96%	341.6	15.2%	358.5	21.0%
Mk06	160.1	233	45.53%	246.1	53.7%	332.3	107.6%
Mk07	243.4	339	39.28%	362.2	48.8%	321.1	31.9%
Mk08	873.4	914	4.65%	952.0	9.0%	1050.6	20.3%
Mk09	596.2	720	20.76%	754.4	26.5%	1051.6	76.4%
Mk10	438.2	631	44.00%	652.2	48.8%	918.2	109.5%
Mk11	997.6	1113	11.57%	1145.1	14.8%	1221.8	22.5%
Mk12	830.0	963	16.02%	1010.0	21.7%	1090.7	31.4%
Mk13	737.5	1069	44.95%	1081.9	46.7%	1427.8	93.6%
Mk14	1037.2	1364	31.51%	1408.6	35.8%	1602.1	54.5%
Mk15	692.4	917	32.44%	952.8	37.6%	1407.0	103.2%

V. RESULTS AND DISCUSSION

Firstly, the experimental validation presented in Section IV demonstrates that Strategy I effectively improves the algorithm's stability with better variance performance, while Strategy II significantly improves the quality of initial solutions, thereby accelerating the optimization process and elevating solution quality. Additionally, Strategy III achieves considerable improvements in both convergence speed and solution quality, with only a marginal increase in computational complexity.

Secondly, experimental results on small-scale instances show that the proposed algorithm produces solutions comparable to those obtained by CPLEX solver, while exhibiting superior computational efficiency. For small-scale problems, our algorithm can effectively generate production

scheduling solutions. In large-scale experiments, the proposed algorithm outperforms other classical algorithms in terms of both solution accuracy and quality, demonstrating its effectiveness in solving flexible job shop scheduling problems that consider both setup and transportation times.

VI. CONCLUSION

Frequent production changes seriously impact the productivity of discrete order-driven manufacturing enterprises. This study considers the machine commissioning time caused by production changeovers and the transportation time due to workpiece changeovers on processing machines. A mixed-integer programming model is formulated to minimize the makespan, and an Improved Sparrow Search Algorithm is proposed to solve it. Experimental comparisons with CPLEX, the Gray Wolf Algorithm, and the Genetic Algorithm confirm the algorithm's effectiveness and efficiency. The results demonstrate that the location updating strategy involving an expanded search direction, the ECT-based population initialization tailored to the problem, and the critical-path-based neighborhood search strategy proposed herein significantly enhance both solution efficiency and quality for the sparrow search algorithm. Future research could refine this study by incorporating employee resource constraints in machine commissioning and the operation of transport equipment to address more complex flexible job shop scenarios.

Future studies will advance this work by integrating machine-setup constraints and operator resource limitations for material-handling equipment, thereby refining the scheduling model to accommodate more complex scenarios in flexible job shop environments.

REFERENCES

[1] DEFERSHA F M, OBIMUYIWA D, YIMER A D. Mathematical model and simulated annealing algorithm for setup operator constrained flexible job shop scheduling problem[J]. Computers & Industrial Engineering, 2022, 171: 108487.

[2] LI J, LIU Z, LI C, et al. Improved artificial immune system algorithm for type-2 fuzzy flexible job shop scheduling problem[J]. IEEE Transactions on Fuzzy Systems, 2020, 29(11): 3234-3248.

[3] PENG Z, ZHANG H, TANG H, et al. Research on flexible job-shop scheduling problem in green sustainable manufacturing based on learning effect[J]. Journal of Intelligent Manufacturing, 2022, 33(6): 1-22.

[4] ZHANG G H, ZHU B Y, YANG Y Y, et al. Research on Flexible Job Shop Scheduling Considering Adjustment Time[J]. Modular Machine Tool & Automatic Manufacturing Technique, 2019(8): 152-156.

[5] SADRZADEH A. Development of Both the AIS and PSO for Solving the Flexible Job ShopScheduling Problem[J]. Arabian Journal for Science and Engineering, 2013, 38(12):3593-3604.

[6] ZHANG Q, MANIER H,MANIER M A. A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times[J]. Computers & Operations Research, 2012, 39(7): 1713-1723.

[7] AN Y, CHEN X, ZHANG J, et al.A hybrid multi-objective evolutionary algorithm to integrate optimization of the production scheduling and imperfect cutting tool maintenance considering total energy consumption[J]. Journal of Cleaner Production, 2020, 268: 121540.

[8] LI J, DENG J, LI C, et al. An improved Jaya algorithm for solving the flexible job shop scheduling problem with transportation and setup times[J]. Knowledge-Based Systems, 2020, 200: 106032.

[9] ZHANG H, XU G, PAN R, et al. A novel heuristic method for the energy-efficient flexible job-shop scheduling problem with sequence-dependent set-up and transportation time[J]. Engineering Optimization, 2022, 54(10): 1646-1667.

[10] SUN J, ZHANG G, LU J, et al. A hybrid many-objective evolutionary algorithm for flexible job-shop scheduling problem with transportation and setup times[J]. Computers & operations research, 2021, 132: 105263.

[11] ROSSI A. Flexible Job Shop Scheduling with Sequence-Dependent Setup and Transportation Times by Ant Colony with Reinforced Pheromone Relationships[J]. International Journal of Production Economics, 2014, 153: 253-267.

[12] ZHANG G, HU Y, SUN J, et al. An improved genetic algorithm for the flexible job shop scheduling problem with multiple time constraints[J]. Swarm and Evolutionary Computation, 2020, 54: 100664.

[13] SHEN L, DAUZÈRE-PÉRÈS S, NEUFELD J S. Solving the flexible job shop scheduling problem with sequence-dependent setup times[J]. European Journal of Operational Research, 2018, 265(2): 503-516.

[14] WANG L, CAI J, LI M, et al. Flexible job shop scheduling problem using an improved ant colony optimization[J]. Scientific Programming, 2017, 2017: 9016303.

[15] KATO E R R, de Aguiar Aranha G D, Tsunaki R H. A new approach to solve the flexible job shop problem based on a hybrid particle swarm optimization and Random-Restart Hill Climbing[J]. Computers & Industrial Engineering, 2018, 125: 178-189.

[16] XUE J K, SHEN B. A novel swarm intelligence optimization approach: sparrow search algorithm[J]. Systems Science & Control Engineering, 2020, 8(1): 22-34.

[17] LUAN F, LI R, LIU S Q, et al. An Improved Sparrow Search Algorithm for Solving the Energy-Saving Flexible Job Shop Scheduling Problem[J]. Machines. 2022, 10(10): 847.

[18] KACEM I, HAMMADI S, BORNE P. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2002, 32(1): 1-13.

[19] PAL M, MITTAL M L, SONI G, et al. A multi-agent system for FJSP with setup and transportation times[J]. Expert Systems with Applications, 2023, 216: 119474.