TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 03/24 pp1283–1299
DOI: 10.26599/TST.2023.9010098
Volume 29, Number 5, October 2024

Cooperative-Guided Ant Colony Optimization with Knowledge Learning for Job Shop Scheduling Problem

Wei Li, Xiangfang Yan, and Ying Huang*

Abstract: With the advancement of the manufacturing industry, the investigation of the shop floor scheduling problem has gained increasing importance. The Job shop Scheduling Problem (JSP), as a fundamental scheduling problem, holds considerable theoretical research value. However, finding a satisfactory solution within a given time is difficult due to the NP-hard nature of the JSP. A co-operative-guided ant colony optimization algorithm with knowledge learning (namely KLCACO) is proposed to address this difficulty. This algorithm integrates a data-based swarm intelligence optimization algorithm with model-based JSP schedule knowledge. A solution construction scheme based on scheduling knowledge learning is proposed for KLCACO. The problem model and algorithm data are fused by merging scheduling and planning knowledge with individual scheme construction to enhance the quality of the generated individual solutions. A pheromone guidance mechanism, which is based on a collaborative machine strategy, is used to simplify information learning and the problem space by collaborating with different machine processing orders. Additionally, the KLCACO algorithm utilizes the classical neighborhood structure to optimize the solution, expanding the search space of the algorithm and accelerating its convergence. The KLCACO algorithm is compared with other high-performance intelligent optimization algorithms on four public benchmark datasets, comprising 48 benchmark test cases in total. The effectiveness of the proposed algorithm in addressing JSPs is validated, demonstrating the feasibility of the KLCACO algorithm for knowledge and data fusion in complex combinatorial optimization problems.

Key words: Ant Colony Optimization (ACO); Job shop Scheduling Problem (JSP); knowledge learning; co-operative guidance

1 Introduction

The research level of a fundamental dispatching theory considerably influences the effectiveness of industrial manufacturing applications. A proficient task scheduling scheme can optimize the utilization of existing production resources, thereby enhancing the enterprise production efficiency. Job shop Scheduling Problem (JSP) is a conventional scheduling problem that has been established as a strongly NP-hard problem[1]. Generating an optimal solution using exact algorithms within the polynomial time is difficult, as proven by previous research[2]. The number of solutions for a JSP with n jobs and m machines is $(n!)^m$. The vast solution space of JSP severely affects

• Wei Li and Xiangfang Yan are with School of Information Engineering, Jiangxi University of Science and Technology, Ganzhou 341000, China. E-mail: liwei@jxust.edu.cn; yanxf@jxust.edu.cn.
• Ying Huang is with School of Mathematics and Computer Science, Gannan Normal University, Ganzhou 341000, China. E-mail: nhwshy@whu.edu.cn.
* To whom correspondence should be addressed.
Manuscript received: 2023-06-06; revised: 2023-08-16; accepted: 2023-09-07

© The author(s) 2024. The articles published in this open access journal are distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/).

the determination of the optimal solution for the problem.

JSP has several variants that are influenced by the characteristics of enterprises due to the nature of industries. These variants include distributed JSP that requires multiple cooperation[3], flexible JSP that considers the same type of processing machines[4], flow shop scheduling problem that involves mass production of the same type[5], and fuzzy JSP that deals with dynamic environments[6]. Numerous real-world application scenarios have contributed to the advancement and development of JSPs, leading to the development of several effective scheduling optimization techniques.

The objective of the JSP is to find a well-considered scheduling plan for a given number of jobs to be executed on a set of machines within a limited time. The current research approach primarily focuses on two directions: model-based improvement and data-driven evolution, with the aim of achieving optimal scheduling. Following the completion of problem modeling, researchers enhance and adapt the model based on practical requirements and subsequently employ intelligent and efficient algorithms to address the respective submodels. Yan et al.[7] strengthened the JSP model by establishing a connection between integer and binary variables in the job shop scheduling model. Yao et al.[8] refined and simplified the mathematical model for JSP by drawing insights from practical robot application scenarios, thereby obtaining superior problem solutions. The advancement of the JSP model has increased the potential for finding the optimal solution for the algorithms. However, researching enhanced models by employing exact algorithms remains impossible within the given temporal constraints.

In contrast to model-based optimization techniques, data-driven evolutionary optimization algorithms offer a dynamic approach to enhancing the quality of scheduling solutions. These algorithms continuously refine the scheduling outcomes by meticulously evaluating the advantages and disadvantages of generated solutions based on predetermined inheritance rules. Sahman et al.[9] used the emerging artificial algae algorithm to solve JSP and achieved excellent results. Mahmud et al.[10] utilized the differential evolution algorithm based on multi-operator communication to find the optimal solution for most JSP benchmark test cases. Xie et al.[11] employed the genetic algorithm combined with tabu search to complete and update the optimal scheduling scheme for some classic JSP benchmark test cases. For JSP with machine flexibility, Xing et al.[12] introduced a knowledge-based Ant Colony Optimization (ACO) algorithm to effectively integrate population information with problem-specific knowledge, enabling a highly efficient spatial search for problem-solving compared to traditional algorithms. The particle swarm optimization, chemical reaction optimization, iterative greedy, and migrating birds optimization algorithms have exhibited good results for JSP and its variants[13–16]. Additionally, a combination of the reinforcement learning method and evolutionary algorithm has yielded exceptional outcomes. Wang et al.[17] demonstrated that reinforcement learning has advantages over other algorithms in dynamic scheduling scenarios. The reinforcement learning algorithm could improve the production of subsequent scheduling schemes by continuously learning and updating previously acquired information. Xi and Lei[18] combined the Q-learning and metaheuristic algorithms to solve the distributed hybrid flow shop problem. Zhao et al.[19] proposed a reinforcement learning based collaborative metaheuristic algorithm to address the energy-saving distributed flow shop scheduling problem. Wu et al.[20] indicated that the real-time scheduling method in JSP outperforms other single scheduling rules when optimizing most objectives. Compared with other intelligent optimization algorithms, the ACO algorithm uses a job-by-job approach to determine the processing order for constructing the scheduling plan, increasing its suitability for real-time scheduling. Furthermore, utilizing the pheromone mechanism in the ACO algorithm provides it with excellent information collection and knowledge utilization capabilities. Therefore, this paper adopts the ACO algorithm as the fundamental algorithmic framework.

The algorithm considers using scheduling rules as the primary criteria for selecting the order of operations to optimize the scheduling scheme effectively. J. Wang and L. Wang[21] used the heuristic scheduling knowledge and historical information generated during the search process to jointly guide the evolutionary search of the algorithm and find the efficient solution to the distributed flow shop scheduling problem. Pan et al.[22] combined various heuristic rules with a

bipopulation evolutionary algorithm to obtain an efficient solution for energy-saving fuzzy flexible JSP. Chen et al.[23] integrated scheduling rules with deep reinforcement learning to structure the job processing sequence and achieve optimal solutions for flexible JSP. The utilization of scheduling rules ensures the quality of the algorithm solution. The scheduling information, which changes in real time during the solution construction process, is considered domain knowledge in JSP and plays a role in the node migration of the ACO algorithm.

Recent studies have confirmed that employing solution-based deep neighborhood search methods can effectively reduce the computational burden of solving problems using an exact algorithm. Zhang et al.[24] and Xie et al.[11] successively proposed neighborhood structures based on critical blocks for JSP critical paths. Their findings indicate that optimizing the neighborhood structure based on critical blocks can considerably improve the quality of heuristic algorithms for solving JSPs. The difference in the neighborhood structure determines the direction and quantity of search branches, affecting the search efficiency and quality. Using a neighborhood-based local search without considering the time consumption of the algorithm can often yield the optimal solution to the problem.

A co-operative-guided ACO algorithm with knowledge learning (namely KLCACO) is proposed in this paper to solve JSP to minimize makespan. KLCACO comprises the following three parts: a framework for individual construction based on knowledge, an adaptive pheromone control mechanism, and a neighborhood-based solution quality improvement strategy. Herein, the knowledge used for individual construction is the dynamic heuristic information based on jobs, which is used for generating scheduling schemes. In the construction process, domain knowledge and pheromones change nodes in different states. The existing pheromone mechanism is improved to increase the suitability of the ACO algorithm for solving JSPs. In addition, the N7 neighborhood is applied to the local optimization strategy to further improve the quality of the algorithm solution. The main contributions of the paper are summarized as follows:

(1) A solution construction scheme based on scheduled knowledge learning is proposed. Multiple operation priorities are used as heuristic factors for constructing the solution for learning real-time scheduling information.

(2) A pheromone guidance mechanism based on a collaboration strategy is introduced. The machine-based operation sequence is stored in the pheromone as population learning information for storage and utilization of this sequence.

(3) A classical neighborhood structure is employed to optimize the local solution. Potential neighborhood optimization solutions are generated by adjusting the order of operations within critical blocks.

(4) The experiments show that the KLCACO algorithm is more efficient than other intelligent optimization algorithms. Therefore, this paper presents a high-quality swarm intelligence optimization algorithm for JSP by combining data- and model-driven approaches.

The remainder of this paper is organized as follows: Section 2 introduces the basic theory of the ACO algorithm and JSP. Section 3 comprehensively describes the KLCACO algorithm proposed in this paper. Section 4 shows the experimental results and their analyses. Section 5 concludes the paper.

2 Related Work

2.1 ACO

The ACO algorithm, originally introduced by Dorigo in 1991, is a swarm intelligence optimization algorithm that draws inspiration from the foraging behavior of ants in natural settings[25]. The fundamental principles of ACO include the construction mechanism of the solution and the subsequent pheromone update process conducted by ants. An illustrative instance of the Ant System (AS) algorithm is presented below, serving as the most fundamental representation of the ACO algorithm[26]. In the AS algorithm, when ant k is located at node i, the permissible set of potential succeeding nodes is denoted as allowSet. Ant k uses the following equation to calculate the probability $\mathrm{Prob}_{i,j}^{k}$ of its transfer from node i to node j:

$$\mathrm{Prob}_{i,j}^{k}=\begin{cases}\dfrac{\tau_{i,j}^{\alpha}\times\eta_{i,j}^{\beta}}{\sum\limits_{k\in\mathrm{allowSet}}\tau_{i,k}^{\alpha}\times\eta_{i,k}^{\beta}}, & j\in\mathrm{allowSet};\\ 0, & \mathrm{other}\end{cases} \tag{1}$$

where $\tau_{i,j}$ and $\eta_{i,j}$ denote the concentration of pheromones and heuristics, respectively, between nodes i and j. α and β are the weights of the

pheromone and heuristics, respectively, in path construction. Ants construct individual solutions using Eq. (1) and then release the pheromone according to the quality of each solution. The corresponding pheromone release equation for the AS algorithm is as follows:

$$\tau_{i,j}^{\text{update}}=\begin{cases}\tau_{i,j}^{\text{old}}+\dfrac{Q}{\text{fit}_k}, \langle i,j\rangle\in \text{sol}_k;\\ \tau_{i,j}^{\text{old}}, \text{other}\end{cases} \tag{2}$$

where $\tau_{i,j}^{\text{update}}$ and $\tau_{i,j}^{\text{old}}$ are the concentrations of pheromone before and after the increase in pheromone on edge $\langle i, j\rangle$, respectively. Q is a constant of fixed size, and the fitness function value for ant k is denoted as fit_k. When all individuals of the ant colony have completed the release of pheromones, pheromone evaporation is conducted. Correspondingly the pheromone concentration on edge $\langle i, j\rangle$ decreases to $\tau_{i,j}^{\text{new}}$,

$$\tau_{i,j}^{\text{new}}=(1-\rho)\tau_{i,j}^{\text{old}} \tag{3}$$

where the evaporation rate is represented by ρ.

Scholars have made extensive contributions to ongoing research on the ACO algorithm due to its remarkable efficacy in addressing combinatorial optimization problems. Therefore, the Ant Colony System (ACS)[27] algorithm and MAX-MIN AS (MMAS)[28] algorithm are proposed as notable advancements. In comparison to the AS algorithm, the ACS and MMAS algorithms increase the influence of elite individuals on the population search direction by intensifying the rate of pheromone release from these exceptional individuals. Notably, the ACS algorithm realizes the aforementioned phenomenon by enhancing the local pheromone update while reducing the global update of pheromones by regular individuals. Conversely, the MMAS algorithm achieves a balanced exploration capability by incorporating a smaller evaporation factor. The global pheromone update equations for the ACS and MMAS algorithms are shown in Eq. (4), and Eq. (6) represents the local pheromone update for the ACS algorithm,

$$\tau_{i,j}^{\text{new}}=\begin{cases}(1-\rho)\tau_{i,j}^{\text{old}}+\rho\times\Delta\tau_{\text{best}}, \langle i,j\rangle\in \text{sol}_{\text{best}};\\ (1-\rho)\tau_{i,j}^{\text{old}}, \text{other}\end{cases} \tag{4}$$

$$\Delta\tau_{\text{best}}=\frac{1.0}{\text{fit}_{\text{best}}} \tag{5}$$

$$\tau_{i,j}^{\text{locNew}}=(1-\vartheta)\tau_{i,j}^{\text{old}}+\vartheta\times\tau_0 \tag{6}$$

where ρ and ϑ are global and local pheromone evaporation factors, respectively. sol_{best} is the optimal scheduling scheme obtained by the ant colony. fit_{best} is the corresponding fitness value, and it is utilized to calculate the increase in pheromone ($\Delta\tau_{\text{best}}$) for this round. τ_0 is a constant employed to control pheromone concentration.

The ACS algorithm uses the pseudorandom state transition rule to improve the possibility of selecting nodes with high probability. When the generated random number q ($q\in[0, 1]$) is less than q_0 (a constant), the ant jumps directly to the node with the maximum probability of transfer; otherwise, the state transition is performed according to Eq. (1),

$$j=\begin{cases}\max\{\text{Prob}_{i,k}^{\text{ant}}, k\in\text{allowSet}\}, q\leqslant q_0;\\ \text{Equation (1)}, \text{other}\end{cases} \tag{7}$$

The MMAS algorithm not only improves the pheromone release strategy of the AS algorithm, but also sets the upper $\tau_{\max}$ and lower $\tau_{\min}$ thresholds for pheromone. In addition, the MMAS algorithm will achieve a local optimal jump through pheromone re-initialization when the algorithm reaches stagnation.

2.2 JSP

The standard model for the JSP can be represented as a quaternion $\langle n, m, J, B\rangle$. n and m are the numbers of jobs and machines involved in processing, respectively, J represents the JSP for scheduling problems, and B is the performance indicator for the scheduling scheme to be solved. The following provides a description of the JSP in terms of its mathematical and graph network models.

2.2.1 Based model

The mathematical model of JSP is introduced, considering the various components involved. Job i comprises c_i operations: $J_i=\{O_{i,1}, O_{i,2}, \ldots, O_{i,c_i}\}$. The processing time of operation $O_{i,k}$ on machine M_t is $T_{i,k,t}$, and the start processing time of operation $O_{i,k}$ is $S_{i,k}$. The binary variable $O_{i,k,t}$ takes the value of 1 if operation $O_{i,k}$ can be processed on machine M_t; otherwise, it is 0. All jobs have equal priority, and no sequential constraints exist among them. The job processing procedure adheres to the following set of constraints:

$$S_{i,k}\geqslant 0,\ T_{i,k,t}\geqslant 0 \tag{8}$$

$$\sum_{c=1}^{m}O_{i,k,c}=1 \tag{9}$$

$$S_{i,k} \geqslant S_{i,k-1} + T_{i,k-1} \tag{10}$$

$$\begin{aligned} &S_{i,k} \geqslant S_{q,l} + T_{q,l,t}, \\ &O_{i,k,m} = O_{q,l,m} = 1, \\ &S_{i,k} \geqslant S_{q,l} \end{aligned} \tag{11}$$

Equation (8) indicates that the start processing time and processing time of each operation for all jobs are positive numbers. Equation (9) indicates that each operation has a unique optional processing machine in the JSP. Formula (10) points to successive constraints on different operations of the same job. Furthermore, Formula (11) implies that when operation $O_{i,k}$ is processed on the same machine as operation $O_{q,l}$, operation $O_{q,l}$ precedes operation $O_{i,k}$; therefore, operation $O_{i,k}$ must wait for the completion of operation $O_{q,l}$ to start processing, suggesting that a machine can complete only up to one operation at a time.

The JSP objective function with the minimum-maximum completion time as the scheduling performance indicator is established as follows:

$$C_{\max} = \min\left\{S_{i,c_i} + T_{i,c_i,t}\right\},\ i = 1, 2, \ldots, n \tag{12}$$

2.2.2 Based graph

Each JSP can be represented using a disjunctive graph: $G = \{N, C, D\}$. Let N denote the set of nodes, encompassing operation nodes, zero-time start node (S), and end-time node (E). C is the set of directed arcs, while D represents a set of disjunctive undirected arcs. C refers to the sequential priority relationship between operations associated with a particular job, where processes within the same job are interconnected in the order of execution. D provides a sequential, undirected edge representation of operations conforming to machine requirements. Additionally, operations executed on the same machine are bidirectionally connected in pairs.

Figure 1 shows a sample JSP for three jobs and three machines. All circular nodes in the diagram form a node set N, black directed edges form a sequence-dependent set C, and dashed lines of different colors represent the sequence-dependent relationships of the operations to be processed on different machines. Green, orange, and blue connected nodes represent the completion of processing on Machines 1, 2, and 3, respectively. The numbers near the nodes indicate the time consumed by each process.

The completion time of JSP, $C_{\max}$, is equal to the time corresponding to the longest acyclic sequence from node S to node E. This sequence is also known as the critical path. The operations on the critical path are called critical operations. On the critical path, operations that belong to the same machine and are adjacent to each other form the critical blocks[11].

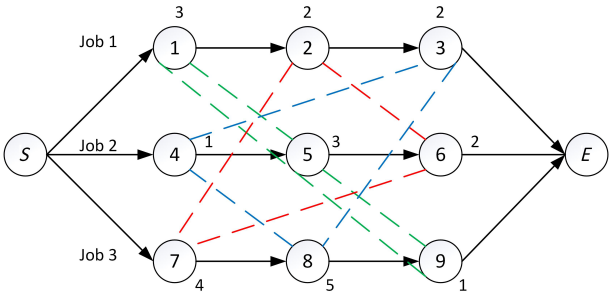

Fig. 1 Sample JSP based on graph.

2.2.3 Representation of JSP solution

At present, most intelligent optimization algorithms choose a serial operation sequence to represent the JSP solution[29, 30], resulting in numerous identical solutions with different representations. For example, in Fig. 2, the blue text below each node is the start processing time of the operation, and a dark node represents a critical operation.

This scheme comprises eight critical operations, three critical blocks, and one critical path. The following two serial processing sequences correspond to the scheduling scheme in Table 1.

Falkenauer and Bouffouix[31] provided solutions to the JSP by recording the processing orders on different machines. The corresponding scheduling scheme is derived when the processing orders of operations on different machines in Fig. 2 are determined. Table 2

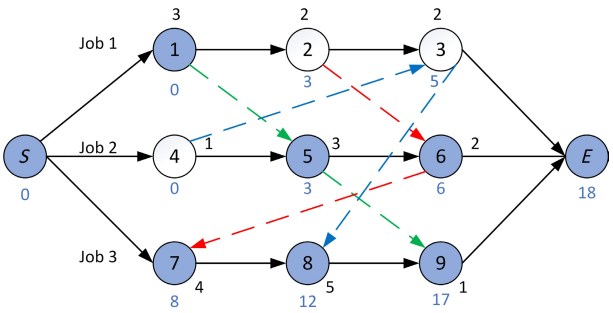

Fig. 2 Sample JSP scheduling scheme based on graph.

Table 1 Different serial representations of the scheduling scheme.

Permutation	Permutation of process nodes
1	*S*-1-4-2-3-5-6-7-8-9-*E*
2	*S*-4-1-5-2-6-7-3-8-9-*E*

Table 2 Parallel expression of the scheduling scheme.

Machine	Permutation of operation	Permutation of job
1	1-5-9	1-2-3
2	4-3-8	2-1-3
3	2-6-7	1-2-3

shows the sequence of processing operations on each machine. Combined with the work constraints of different operations, the JSP disjunctive map can be double-mapped with the solution based on the processing orders on different machines. Thus, the corresponding scheduling scheme and its critical path can be obtained by determining the processing order between the operations on the same machine.

2.2.4 Scheduling scheme for JSP solution

Sequential[32], active, semi-active, and no-delay scheduling can be used for a given machine process schedule. Pinedo proved that active scheduling is the optimal scheduling method for the scheduling problem, with the maximum completion time as the performance indicator[33].

2.3 Motivation of this work

As one of the most standard combinatorial optimization problems, JSP exhibits NP-hard characteristics. Among the existing intelligent optimization algorithms, the ACO algorithm is the most suitable for solving combinatorial optimization problems and has been successfully applied in numerous instances[34–36]. Furthermore, the real-time scheduling method outperforms other single scheduling rules for optimizing most of the objectives. The node-by-node construction mechanism of the ACO algorithm enhances its potential to find the optimal solution for the objectives. Therefore, the ACO algorithm is selected as the fundamental algorithmic framework.

Heuristic information that changes in real time during the scheduling process is employed as problem domain knowledge to ensure the quality of the algorithm solutions, and heuristic information actively participates in the node selection of ants. The dynamic change in JSP knowledge can offer improved environmental information for the algorithm. In this paper, the pheromone mechanism of the ACO algorithm was adaptively updated. The traditional single pheromone matrix cannot simultaneously balance the difference in pheromone concentration on different machines during processing operations. Therefore, a machine-based compound pheromone mechanism is proposed to reduce the significant information difference between various machines. The pheromone interference is helpful for improving the algorithm solution quality. In addition, a classic neighborhood structure is employed to enhance the quality of local solutions generated with the algorithm. The selection of the N7 neighborhood considers the solution performance and number of neighborhoods to ensure optimal algorithm performance.

3 Proposed Approach

The ACO algorithm generates solutions by combining prior search information with existing knowledge and selects the ant search path based on pheromone adjustments. In this paper, the ACO algorithm is integrated with the JSP structure, leading to an improved solution construction mechanism. Additionally, the mode of pheromone modification is adjusted, and a suitable optimization strategy is employed for the neighborhood structure to optimally extract the local information. The algorithm ultimately finds an efficient solution for JSP.

3.1 Algorithm framework

The complete pseudocode of the algorithm is presented in Algorithm 1. The KLCACO algorithm comprises three main stages: initialization of population information, construction of feasible solutions based on knowledge, and updation of the population and knowledge through pheromone adjustment.

In the first stage, the KLCACO algorithm initializes the parameters (Line 1). Subsequently, the greedy construction method is used to generate greedy solutions (Line 2), and the initial pheromone thresholds τ_{limit} are obtained from the greedy solution (Line 3).

During the construction phase of feasible solutions, each ant generates individual solutions using a solution construction mechanism based on scheduling knowledge learning (Lines 5 and 6). The pheromone threshold is adjusted in real time according to the global optimal solution when the local optimal solution for the current round is recorded (Lines 8−10 and 15−17). If all ants in this round complete their individual solution construction, then the local update strategy based on neighborhood structure is used to generate neighborhood optimal solutions, which further improves the quality of all optimal solutions (Lines 13−17).

When the quality of the updated local optimal

Algorithm 1 KLCACO algorithm framework

Input: Information for operations, ant colony size: antPop, pheromone threshold: τ_{limit}, solution of global best: x_{gb}, solution of local best: x_{lb}, solution of neighborhood best: x_{nb}, fitness of function values for ant (fit_{ant}), forglobal best (fit_{gb}), and local best (fit_{lb}), similarity threshold: simi, algorithmic stagnation ceiling: maxStag, algorithmic stagnation counter: stagnateTime

Output: solution of global best: x_{gb}

```
1: Parameter initialization;
2: Calculate greedy solution x_gb;
3: Set τ_limit according to fit_gb;
4: while the termination criterion is not met do
5:     for ant in antPop do
6:         Production scheduling plan x_ant using knowledge
               based construction mechanism;
7:         Update x_lb based on x_ant;
8:         if fit_lb < fit_gb then
9:             x_gb = x_lb;
10:            Update τ_limit according to fit_gb;
11:        end if
12:    end for
13:    Generate x_nb according local optimization with x_lb;
14:    Update x_lb based on x_nb;
15:    if fit_lb < fit_gb then
16:        x_gb = x_lb;
17:        Update τ_limit according to fit_gb;
18:    else
19:        Calculate the similarity s between x_lb and x_gb;
20:        if s > simi then
21:            stagnateTime++;
22:        end if
23:    end if
24:    Update pheromone according to x_gb or x_lb;
25:    if stagnateTime > maxStag then
26:        Pheromone re-initialization based on x_gb;
27:    end if
28: end while
```

solution is lower than that of all the other optimal solutions, the individual similarity between the local and global optimal solutions is calculated (Line 19), and the stagnation time (local exploration time) of the algorithm is increased when the similarity threshold is met (Lines 20–22). The pheromone is subsequently updated using the optimal solution (Line 24). When the number of stops reaches the predetermined upper limit, the KLCACO algorithm uses the globally optimal individual to re-initialize the pheromone to avoid the stoppage state (Lines 25–27). The above operations are completed in the information update and pheromone adjustment stages.

3.2 Knowledge-based construction mechanism

In the context of machine-based operation selection, the ACO algorithm utilizes heuristics and pheromones to determine the subsequent scheduling. The selection of heuristic factors influences the dependence of the algorithm's solution generation on the available information categories. A construction mechanism founded on knowledge learning is introduced in this study, which combines local scheduling rules and the pheromone guidance mechanism. The dynamically changing operation information in the JSP scheduling process participates in the construction of the ant's solution as knowledge.

This study establishes a heuristic factor selection scheme based on compound priority scheduling rules to counteract the defects introduced by single-priority scheduling rules. The heuristic strategy groups are the Earliest Starting Time (EST), the Earliest Finishing Time (EFT), the Longest Wait Time (LWT), and the Longest Residual Time (LRT). The scheduling knowledge is disturbed by random selection to enhance the diversity of the algorithm. The calculation method and selection motivation are shown in Table 3.

During the solution construction process of an individual, the initial operation of each job is categorized based on the corresponding processing machine. A roulette selection scheme is employed to

Table 3 Calculation methods and selection motivations for different scheduling rules.

Heuristic strategy	Calculation method	Motivation
EST	Calculate start time of each operation	An early start processing time can improve equipment utilization efficiency.
EFT	Calculate end time of each operation	A fast completion time can quickly reduce the number of waiting processing operations.
LWT	Calculate waiting time of each operation	Avoid falling into a hungry state due to an operation, which significantly increases the system's time consumption.
LRT	Calculate remaining time of each operation	Increase the processing priority of tasks with longer durations to avoid potential long waiting times.

assign heuristics to machines for hosting processing operations,

$$\psi = \begin{cases} \text{EST}, & 0 \leqslant p \leqslant p_1; \\ \text{EFT}, & p_1 < p \leqslant p_2; \\ \text{LWT}, & p_2 < p \leqslant p_3; \\ \text{LRT}, & p_3 < p \leqslant 1 \end{cases} \tag{13}$$

The scheduling knowledge is determined by finding the position of the generated random number p. The obtained heuristic is then normalized within the range $[\tau_{\min}, \tau_{\max}]$ to eliminate inherent disparities between the heuristic and pheromone.

Equation (14) is employed to compute the selection probability of each operation,

$$\text{Prob}_j = \tau_{i,j}^{\alpha} \times \eta_j^{\beta}, \ j \in \text{actSet}\,(m) \tag{14}$$

$$j = \begin{cases} \max\{\text{Prob}_k, \ k \in \text{actSet}\}, & q \leqslant q_0; \\ \text{Equation (14)}, & \text{other} \end{cases} \tag{15}$$

where i refers to the node of the previous operation with process j on the machine, and actSet (m) represents the set of optional processing operations available on machine m at the current time. Figure 3 shows the basic flow of the KLCACO algorithm to construct a solution.

After operation selection, the resultant selection scheme is stored within the machine's processing order queue. When all the executable machines in this round have finished the process assignment, the subsequent operations that complete the process are added to the candidate processing sequence of the corresponding machine. The above process is followed until all the operations for all the jobs are completed, and the scheduling plan is then stored in the complete processing queue of the machine.

Before starting the scheduling process, the KLCACO algorithm (Algorithm 2) initializes the set of optional operations based on the machines, actSet_i, and the queue of completed operations, FinSet_i, as empty. Lines 1–3 of the pseudocode show that initial operations for each job are stored in the actSet corresponding to the respective machine. The algorithm begins job processing from Line 4, with each machine simultaneously processing available operations to address (Lines 4–15). If the actSet_m is empty, then the processing for that machine m is skipped in the current round until operations are stored in its actSet_m (Lines 6–8). The KLCACO algorithm first utilizes Eq. (13) to determine the scheduling knowledge as a heuristic factor for this iteration (Line 9) and selects the next operation to be processed on a machine. The heuristic factors for each operation are subsequently calculated based on Table 3 (Lines 10–12). These heuristic factors are normalized to eliminate dimensional differences and combined with the relevant pheromone concentration on the edges to obtain the selection probability for each operation in the current round (Line 13). The final operation selection for this processing step is created using Eqs. (14) and (15) via a pseudorandom roulette wheel selection mechanism (Line 14). After determining the processing operations, they are stored in the finSet of the corresponding machine and adjacent operations are added to the actSet based on the job sequence

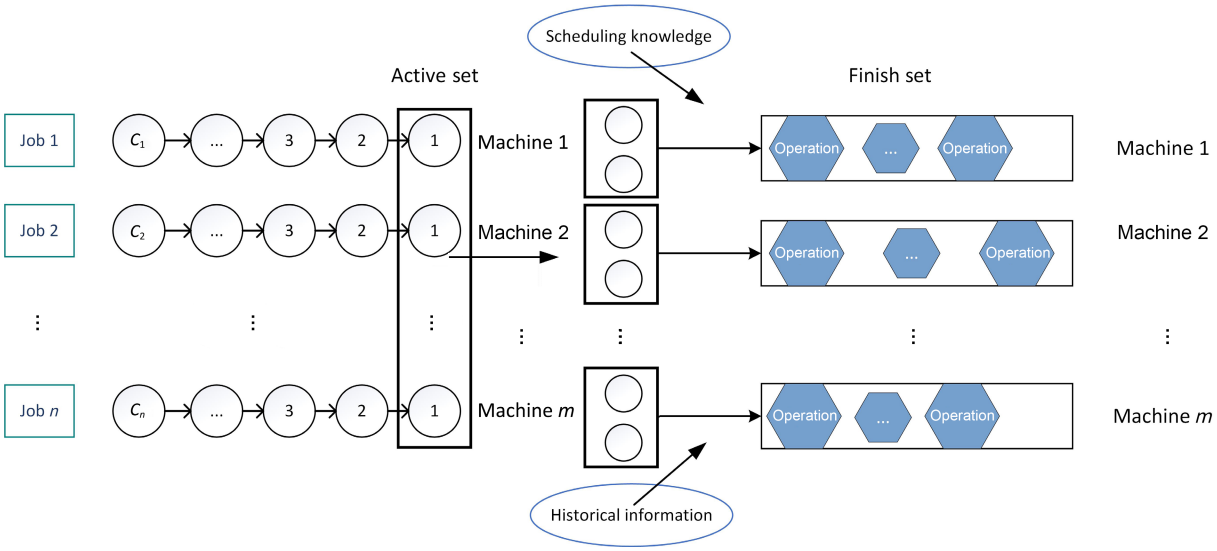

Fig. 3 Solution construction scheme based on knowledge learning.

Algorithm 2 KLCACO algorithm construct solution

Input: Job set: J_n, machine set: M_m, information of operate, operation activity queue actSet_i and completion queue FinSet_i of the i-th machine, list of next waiting processing operations for each job: currLocJob

Output: JSP scheduling plan (FinSet)

1: **for** j in J_n **do**
2: $O_{j,1}$ operation is added to the corresponding actSet;
3: **end for**
4: **while** not finish scheduling **do**
5: **for** m in M_m **do**
6: **if** actSet_m is empty **then**
7: SK_{ip} processing of current machine;
8: **end if**
9: Determine heuristic factors by Eq. (13);
10: **for** operate in actSet_m **do**
11: Calculate heuristic factor according to Table 3;
12: **end for**
13: Normalize heuristic factors and calculate the selection probability of the operation;
14: Complete operation selection on machine m by Eqs. (14) and (15);
15: **end for**
16: Update actSet and FinSet based on currLocJob;
17: **end while**

constraints until all operations are completed (Line 16).

During the solution construction, different heuristic information that changes in real time is integrated and used as scheduling knowledge to facilitate participation in the path generation process. Diverse knowledge is selected through Equation (13), and knowledge-based node transfer is completed with Eqs. (14) and (15), thereby realizing the comprehensive utilization of scheduling knowledge in the ant solution construction process. In addition, this method can maintain the relevant queue in real time with the construction of ants without increasing the time complexity in the deconstruction process and exhibits good adaptability.

3.3 Adaptive pheromone control

3.3.1 Pheromone representation

Within the context of the ACO algorithm, the pheromone mechanism serves as a repository for all information acquired during the algorithm's search for a given problem. This mechanism preserves the obtained information, allowing subsequent ants to access the pheromone matrix and effectively leverage preceding knowledge. A machine selection based pheromone representation scheme, comprising m uniform-sized pheromone matrices, is proposed in this study. Important information is encoded as the sequential data pertaining to distinct operations executed on the same machine. Additionally, a global pheromone threshold is established to govern the system.

As depicted in Fig. 4, an exclusive pheromone matrix is assigned to each machine, with its dimensions determined by the number of jobs allocated to that specific machine. This study uses an $(n+1)\times(n+1)$ matrix as a demonstration (where n signifies the total number of jobs) to account for the unique characteristics of the JSP.

3.3.2 Pheromone threshold control

Upon completion of a round solution by the ant colony, the pheromone update process is conducted by employing either the global or local optimal solution, as dictated by

$$\tau_{i,j}^{\text{new}} = \begin{cases} (1-\rho)\tau_{i,j}^{\text{old}} + \Delta\tau_{\text{best}}, \langle i, j\rangle \in \text{Sche}_{\text{best}}; \\ (1-\rho)\tau_{i,j}^{\text{old}}, \text{other} \end{cases} \tag{16}$$

where Sch_{best} represents the selected optimal scheduling scheme.

Jia et al.[37] and Sttzle and Hoos[38] used an approach based on pheromone threshold control using the optimal solution to achieve a balance between exploration and exploitation capabilities within the algorithm. This study employs the information obtained from the global optimal solution to facilitate the adaptive control of the pheromone threshold,

$$\tau_{\max} = \frac{1}{(1-\rho)\times \text{fit}_{\text{gb}}} \tag{17}$$

$$\tau_{\min} = \tau_{\max} \times \frac{\left(1-\text{pr}^{\frac{1}{n_{\text{ps}}}}\right)}{(n_{\text{ps}}/2-1)\times \text{pr}^{\frac{1}{n_{\text{ps}}}}} \tag{18}$$

where n_{ps} represents the population size and pr is a constant, which is set to 0.05. Updates are made to the

Job ID	S	1	2	...	n
S	0	$\tau_{\max}$	$\tau_{\max}$	...	$\tau_{\max}$
1	$\tau_{\max}$	0	$\tau_{\max}$	...	$\tau_{\max}$
2	$\tau_{\max}$	$\tau_{\max}$	0	...	$\tau_{\max}$
⋮	⋮	⋮	⋮	⋱	⋮
n	$\tau_{\max}$	$\tau_{\max}$	$\tau_{\max}$	...	0

Fig. 4 Pheromone representation example.

global optimal solution; thus, the pheromone threshold promptly triggers adjustments to the upper and lower limits of pheromones on each side. Consequently, the variation in pheromone levels between different paths undergoes alteration, accelerating the convergence of the algorithm.

3.3.3 Stagnation detection based on individual similarity

A similarity measure based on the solution scheme is proposed to effectively evaluate the evolution state of the algorithm. The algorithm can determine whether a solution is stagnant or not by calculating the optimal solution similarity under different iteration rounds. When the algorithm is stagnant, the KLCACO algorithm will re-initialize the pheromone with the global optimal solution to jump out of the stagnation state.

Machine-based solutions are then matrixed according to the machine, and the preference relationship between different processes is represented by the values 0−1. As shown in the first line of Solution 1 in Fig. 5, the corresponding operations of Job1 are executed on the machine before those of other jobs.

The two solutions corresponding to the matrix are summarized, and the number of elements is calculated as 2 in the cumulative matrix. The similarity measure equation is as follows:

$$s=\frac{\text{count}_2}{m\times n\times(n-1)/2} \tag{19}$$

where count_2 is the total number of occurrences of element 2 in the cumulative matrix, and $n\times(n-1)/2$ is the number of elements of a solution for a single machine.

The algorithm sets the parameter simi to control the determination of the exploration status. If the similarity of the local optimal solution to the global optimal solution simi is larger than s, then the number of algorithm stops, stagnateTime, is incremented by one. The KLCACO algorithm will then initialize the pheromones through the global optimal solution when the number of algorithm stops reaches the threshold.

3.4 Local optimization method

JSPs have a large solution space. Therefore, using neighborhood-based local solution optimization is necessary. This paper utilizes the N7 neighborhood structure proposed by Zhang et al.[24] to improve the local solution quality. Figure 6 illustrates the N7 neighborhood diagram.

The KLCACO algorithm performs N7-based neighborhood searches for the local optimal solution generated by each iteration of the ant colony. This algorithm then uses the resulting optimal neighborhood solution to replace the local optimal solution generated by the KLCACO algorithm.

3.5 Analysis of the computational complexity

In the KLCACO algorithm, the main calculation processes include ant solution construction, fitness value evaluation of the solution, pheromone update, neighborhood-based local optimization, and adaptive pheromone threshold adjustment. Assuming that the JSP size is N jobs and M machines and ant colony size is set to Pop, the time complexity of the above process is shown in Table 4.

The time complexity required for all ants in the ant colony to complete the solution construction and fitness value calculation is $\text{Pop}\times O(N^2M^2)$. Further, the best individiual takes $O(N^2)$ time to complete the neighborhood-based local update. The time complexity corresponding to the pheromone update and control process is $O(N^2M)+O(1)$. Overall, the time complexity of the KLCACO algorithm is $O(\text{Pop}\times N^2\times M^2)$.

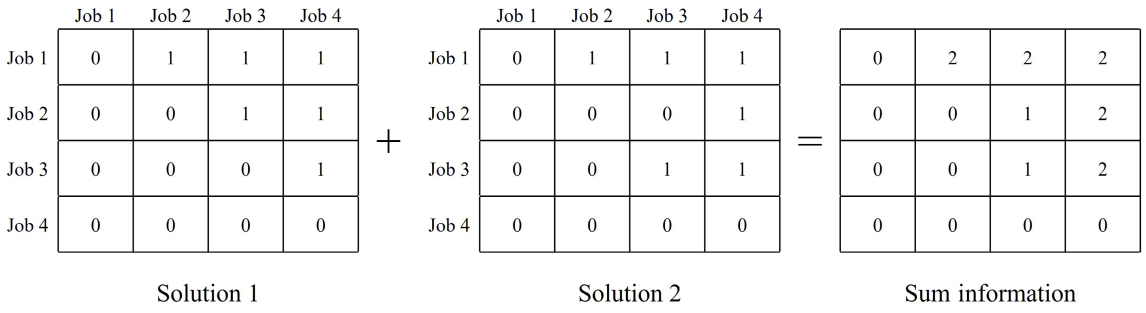

Fig. 5 Similarity calculation between two individuals.

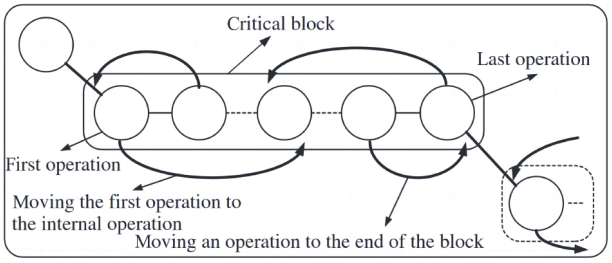

Fig. 6 N7 neighborhood structure.

Table 4 Analysis of algorithm time complexity.

Calculation process	Time complexity
Ant construction solution and evaluation of the fitness	$O(N^2M^2)$
Pheromone update	$O(N^2M)$
Neighborhood-based local optimization	$O(N^2)$
Adaptive pheromone threshold	$O(1)$

4 Experiment and Analysis

A series of experiments is conducted as follows to evaluate the efficacy of the proposed KLCACO algorithm. First, the Taguchi experimental design method is employed to ascertain the optimal combination of parameters related to key aspects of the algorithm. Subsequently, an analysis is performed to evaluate the effectiveness of the three strategies employed by the KLCACO algorithm. Finally, a comparison is conducted between the KLCACO and state-of-the-art algorithms to validate its efficiency and performance. The algorithms are tested on various benchmarks, including the ABZ[39], FT[40], LA[41], and ORB[42] datasets. Table 5 shows the main test sample data used.

All the algorithms were carefully re-implemented in the C++ programming language to ensure a fair comparison. They were executed on an Intel (R) Core (TM) i5-12400F CPU @ 2.50 GHz with 16 GB RAM within the Windows 11 Operating System and Visual Studio 2019.

Table 5 Sample information for job shop scheduling problem.

Best sample	Size ($n \times m$)	Best sample	Size ($n \times m$)
ft06	6 × 6	la06-10	15 × 5
ft10	10 × 10	la11-15	20 × 5
ft20	20 × 5	la16-20	10 × 10
abz5-6	10 × 10	la21-25	15 × 10
abz7-9	20 × 15	la26-30	20 × 10
la01-05	10 × 5	orb01-10	10 × 10

4.1 Optimal parameter combination

The experimental design method of Taguchi[43] is used in this paper to achieve the optimal control of parameters. The parameters of the KLCACO algorithm are categorized based on their relevance to the algorithm to facilitate their efficient configuration. These parameters are divided into two groups: key parameters that control the performance of strategies and general parameters that control the overall algorithm process. The key parameters include the heuristic factor weight β, pseudorandom probability q_0, pheromone evaporation factor ρ, similarity threshold simi and the parameter group of Priority Scheduling Rules (PSR: p_1-p_2-p_3) in a compound heuristic scheme. Meanwhile, the general parameters include the population size pop and pheromone re-initialization threshold maxStag. These parameters influence the evolutionary process of the algorithm. Considering the limited evaluation times, a parameter configuration of pop = 50 and maxStag = 50 is chosen.

Taguchi's experimental design method was employed to match different parameter groups, and the experimental process is presented in Tables 6–9 and Fig. 7. The effectiveness of the solution was evaluated using the average solution quality (Efficacy) obtained from running the algorithm 10 times. Table 6 outlines key parameter variations, specifically identifying three levels (T1–T3) to differentiate PSR parameters. Results of the Taguchi experiment are summarized in Table 7, with Table 8 providing an in-depth analysis. Assessing the impact of different parameters on algorithm performance yields their respective rank. Figure 7 visually represents the influence of key parameters on experimental outcomes. The parameter selection for the KLCACO algorithm proposed in this paper is detailed in Table 9.

4.2 Strategy effectiveness analysis

This section discusses the effectiveness of the three strategies employed in the KLCACO algorithm. The algorithms included in this experimental comparison comprise the traditional MMAS, the KLCACO algorithm, and three variant algorithms. Specifically,

Table 6 Experimental factors and levels of KLCACO.

Level	β	q_0	ρ	simi	PSR
1	0.5	0.5	0.05	0.2	0.25-0.5-0.75 T1
2	1.0	0.6	0.10	0.3	0.3-0.6-0.8 T2
3	2.0	0.7	0.20	0.4	0.2-0.4-0.7 T3

Table 7 Taguchi's experimental design results.

Index	β	q_0	ρ	simi	PSR	Efficacy
1	0.5	0.5	0.05	0.2	T1	1044.3
2	0.5	0.5	0.05	0.2	T2	1038.0
3	0.5	0.5	0.05	0.2	T3	1043.3
4	0.5	0.6	0.10	0.3	T1	1039.8
5	0.5	0.6	0.10	0.3	T2	1034.6
6	0.5	0.6	0.10	0.3	T3	1042.5
7	0.5	0.7	0.20	0.4	T1	1036.9
8	0.5	0.7	0.20	0.4	T2	1022.2
9	0.5	0.7	0.20	0.4	T3	1031.1
10	1.0	0.5	0.10	0.4	T1	1044.6
11	1.0	0.5	0.10	0.4	T2	1042.2
12	1.0	0.5	0.10	0.4	T3	1047.5
13	1.0	0.6	0.20	0.2	T1	1041.4
14	1.0	0.6	0.20	0.2	T2	1039.1
15	1.0	0.6	0.20	0.2	T3	1042.8
16	1.0	0.7	0.05	0.3	T1	1032.8
17	1.0	0.7	0.05	0.3	T2	1028.9
18	1.0	0.7	0.05	0.3	T3	1030.5
19	2.0	0.5	0.20	0.3	T1	1038.6
20	2.0	0.5	0.20	0.3	T2	1039.1
21	2.0	0.5	0.20	0.3	T3	1043.6
22	2.0	0.6	0.05	0.4	T1	1030.0
23	2.0	0.6	0.05	0.4	T2	1034.9
24	2.0	0.6	0.05	0.4	T3	1039.6
25	2.0	0.7	0.10	0.2	T1	1033.2
26	2.0	0.7	0.10	0.2	T2	1024.3
27	2.0	0.7	0.10	0.2	T3	1039.0

Table 8 Analysis of KLCACO test results.

Level	β	q_0	ρ	simi	PSR
1	1037	1042	1036	1038	1038
2	1039	1038	1039	1037	1034
3	1036	1031	1037	1037	1040
Rank	3	1	4	5	2

Table 9 KLCACO parameters setting.

Parameter	Value
α	1.0
β	2.0
q_0	0.7
ρ	0.05
simi	0.4
p_1	0.3
p_2	0.6
p_3	0.8
pop	50

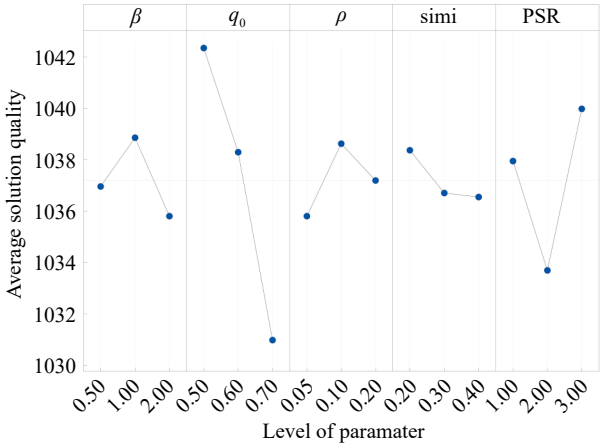

Fig. 7 Main effect diagram of parameters.

the first variant, which is denoted as MMAS+KL, enhances the MMAS algorithm by adopting a knowledge-based construction mechanism. The second variant, which is referred to as MMAS+N7, is a modification of the MMAS algorithm wherein the N7 local search component is added. The third variant, which is designated as MMAS+Phe, improves the solution construction mechanism based on scheduling knowledge learning from the MMAS algorithm. Table 10 shows the results of the strategy effectiveness experiment.

Tested on benchmarks of different sizes from the LA dataset, each algorithm is executed independently for 10 iterations. The parameters of the algorithm are configured using the values obtained in the previous section (Table 9). The best solution (Best), the average solution quality (Efficacy), and the Relative Percentage Deviation (RPD) of the algorithms are calculated. The RPD is computed as follows:

$$\mathrm{RPD}=\frac{\mathrm{Efficacy}-\mathrm{LB}}{\mathrm{LB}}\times 100 \tag{20}$$

where LB is the lowest lower bound for this instance. The minimum relative percentage deviations obtained by the algorithms in Table 10 are highlighted in bold.

The strategies proposed in this paper have led to significant improvements in the solving performance of the algorithm compared to MMAS. Among these strategies, the MMAS+KL algorithm has exhibited remarkable success. This achievement can be primarily attributed to the efficient utilization of information within the dynamic scheduling process facilitated by the knowledge-based construction mechanism. The use of roulette wheel gambling for knowledge selection ensures algorithmic diversity, effectively preventing

Table 10 Experimental results of algorithm strategy effectiveness.

Instance	LB	Size	MMAS			MMAS+KL			MMAS+N7			MMAS+Phe			KLCACO		
			Best	Efficacy	RPD	Best	Efficacy	RPD	Best	Efficacy	RPD	Best	Efficacy	RPD	Best	Efficacy	RPD
la01	666	10×5	678	686.7	3.11	666	666.0	**0.00**	666	670.0	0.60	666	672.4	0.96	666	666.0	**0.00**
la07	890	15×5	906	917.8	3.12	890	892.4	0.27	890	903.1	1.47	890	896.0	0.67	890	890.0	**0.00**
la13	1150	20×5	1156	1160.4	0.90	1150	1150.0	**0.00**	1150	1150.5	0.04	1150	1156.2	0.54	1150	1150.0	**0.00**
la18	848	10×10	940	961.2	13.35	886	897.4	5.83	922	937.5	10.55	885	901.7	6.33	861	884.1	**4.26**
la21	1046	15×10	1237	1259.8	20.44	1160	1181.1	12.92	1214	1242.3	18.77	1180	1195.6	14.30	1134	1164	**11.28**
la26	1218	20×10	1439	1481.0	21.59	1339	1370.7	12.54	1440	1447.3	18.83	1399	1416.3	16.28	1325	1352.5	**11.04**

the convergence of the algorithm to local optima. Additionally, the incorporation of the pheromone mechanism and optimization of the neighborhood structure have further enhanced the capabilities of the MMAS algorithm. The enhanced pheromone mechanism is particularly conducive to the centralized processing of pheromones scattered by ants between different operations on the same machine, resulting in further improvements in the efficiency of the algorithm.

Table 10 shows that the KLCACO algorithm outperforms other improved algorithms in optimal and average solution quality. Therefore, the three strategies proposed in this paper effectively enhance algorithmic performance, and their integration yields superior outcomes in solving JSPs.

4.3 Comparison with other evolutionary algorithms

The KLCACO algorithm is compared with three state-of-the-art algorithms to verify its effectiveness, which are MCDE/TS[10], HA[11], and AAA[9], these have been the top competitors for solving JSP in recent years. These algorithms adopt the optimal parameter combinations proposed by their authors to ensure their optimal performance. The standard deviation (Mean error) is used as an additional evaluation indicator to evaluate the stability of the above algorithms. Table 11 presents the parameters of these algorithms.

All algorithms utilize a consistent maximum of 10 000 fitness evaluation iterations as the termination criterion. Each algorithm undergoes 10 independent runs. The results of the experiments are presented in Table 12. The comparative result for a problem is given as "+/ = /−" in parentheses, which indicates that the RPD of the compared algorithm is better than, equal to, or poorer than KLCACO. The best RPD value for each group is shown in boldface.

Table 12 illustrates that the KLCACO algorithm has

Table 11 Parameter settings for comparison algorithms.

Algorithm	pop	Parameter value
AAA[9]	40	K=2, L_e=0.3, A_p=0.5
HA[11]	30	p_c=0.9, p_m=0.1, α=sd/5, β=max{sd/10, 2}
MCDE/TS[10]	50	C_r=0.9, F=0.9, p=30
KLCACO	50	α=1, β=2, q_0=0.7, ρ=0.05, simi=0.4

demonstrated superior outcomes for the majority of test cases when compared to the AAA and HA algorithms. Comparison results with the MCDE/TS algorithm revealed that the performances of the two algorithms are nearly equivalent. On approximately two-thirds of the data instances, the results obtained by the KLCACO algorithm are better than or equivalent to the MCDE/TS algorithm. The MCDE/TS algorithm, which utilizes tabu search, explores a highly extensive neighborhood space, accounting for the inferior performance of the KLCACO algorithm on the remaining test instances. Overall, the experimental results reveal that the KLCACO algorithm exhibits a notable ability for solving JSP.

The mean standard deviations of the compared algorithms over all samples are shown in Fig. 8. Based on the experimental data, the KLCACO algorithm exhibits substantially greater stability in problem-solving compared to other algorithms, demonstrating a mean standard deviation of 7.58 across all test cases. The utilization of the knowledge-based solution construction mechanism significantly reduces uncertainty due to random number selection during algorithm execution. Simultaneously, the neighborhood search based on individual solutions provides further assurance for the stability of solutions of the algorithm. The consistent and stable results suggest that the KLCACO algorithm performs reliably across various scenarios of JSP samples.

Regarding solving effectiveness, the experimental group did not calculate the lower bound solution for most data samples due to the limitation of the

Table 12 Experimental results for algorithm comparison.

Instance	Size	AAA				HA				MCDE/TS				KLCACO			
		Best	Efficacy	Mean error	RPD	Best	Efficacy	Mean error	RPD	Best	Efficacy	Mean error	RPD	Best	Efficacy	Mean error	RPD
abz5	1234	1329	1367.2	18.24	10.79	1305	1389.5	68.05	12.60	1265	1283.8	10.22	4.04	1261	1273.5	7.27	**3.20**
abz6	943	996	1035.5	20.02	9.81	1004	1075.9	36.59	14.09	947	969.1	11.41	2.77	951	956.3	3.66	**1.41**
abz7	656	843	855.1	6.61	30.35	802	851.3	43.03	29.77	753	780.5	15.92	18.98	754	764.5	5.37	**16.54**
abz8	648	865	880.1	9.92	35.82	828	899.6	38.26	38.83	752	786.2	15.89	**21.33**	782	790.6	3.69	22.01
abz9	678	893	907.1	10.07	33.79	856	902.3	22.13	33.08	783	826.7	22.90	21.93	803	809.7	3.52	**19.42**
ft06	55	55	56.5	1.02	2.73	55	59.3	3.49	7.82	55	55.1	0.30	0.18	55	55.0	0.00	**0.00**
ft10	930	1088	1112.8	15.15	19.66	1032	1173.5	73.26	26.18	1000	1033.5	19.76	**11.13**	1009	1035.6	15.72	11.35
ft20	1165	1343	1366.1	19.55	17.26	1467	1540.9	35.74	32.27	1251	1297.9	43.75	11.41	1271	1293.6	11.89	**11.04**
la01	666	666	670.5	3.58	0.68	666	684.4	21.22	2.76	666	666.0	0.00	**0.00**	666	666.0	0.00	**0.00**
la02	655	686	713.0	12.95	8.85	693	748.4	28.00	14.26	668	686.6	8.43	**4.82**	660	687.1	11.82	4.90
la03	597	623	636.0	8.99	6.53	627	696.2	38.79	16.62	613	631.5	13.70	5.78	619	624.2	3.57	**4.56**
la04	590	590	621.0	12.79	5.25	632	696.4	33.47	18.03	590	600.1	7.22	**1.71**	616	620.1	4.23	5.10
la05	593	593	593.0	0.00	**0.00**	593	593.0	0.00	**0.00**	593	593.0	0.00	**0.00**	593	593.0	0.00	**0.00**
la06	926	926	926.0	0.00	**0.00**	926	929.0	5.69	0.32	926	926.0	0.00	**0.00**	926	926.0	0.00	**0.00**
la07	890	897	912.8	8.54	2.56	918	956.9	41.53	7.52	890	911.8	19.10	2.45	890	890.4	0.92	**0.04**
la08	863	864	873.1	5.66	1.17	863	880.8	21.78	2.06	863	863.0	0.00	**0.00**	863	865.8	4.33	0.32
la09	951	951	951.0	0.00	**0.00**	951	954.3	7.21	0.35	951	951.3	0.90	0.03	951	951.0	0.00	**0.00**
la10	958	958	958.0	0.00	**0.00**	958	959.1	3.30	0.11	958	958.0	0.00	**0.00**	958	958.0	0.00	**0.00**
la11	1222	1222	1222.0	0.00	**0.00**	1222	1225.0	5.02	0.25	1222	1222.0	0.00	**0.00**	1222	1222.0	0.00	**0.00**
la12	1039	1039	1039.0	0.00	**0.00**	1039	1080.6	45.81	4.00	1039	1039.0	0.00	**0.00**	1039	1039.0	0.00	**0.00**
la13	1150	1150	1150.0	0.00	**0.00**	1150	1168.5	32.16	1.61	1150	1150.9	2.70	0.08	1150	1150.0	0.00	**0.00**
la14	1292	1292	1292.0	0.00	**0.00**	1292	1292.0	0.00	**0.00**	1292	1292.0	0.00	**0.00**	1292	1292.0	0.00	**0.00**
la15	1207	1260	1285.8	12.21	6.53	1268	1394.1	91.42	15.50	1207	1238.4	23.75	**2.60**	1246	1256.8	10.26	4.13
la16	945	1011	1045.7	22.33	10.66	999	1079.0	47.85	14.18	946	1004.9	33.47	6.34	989	1002.5	7.39	**6.08**
la17	784	852	872.2	9.00	11.25	819	873.2	53.09	11.38	796	816.0	21.16	**4.08**	802	817.7	6.36	4.30
la18	848	911	944.1	17.16	11.33	901	963.5	41.72	13.62	861	890.6	19.28	5.02	861	883.8	10.48	**4.22**
la19	842	930	957.5	14.07	13.72	936	970.6	35.89	15.27	863	893.9	15.55	6.16	877	886.1	4.70	**5.24**
la20	902	968	994.1	12.47	10.21	973	1018.0	33.11	12.86	907	927.8	22.91	**2.86**	902	935.1	12.33	3.67
la21	1046	1223	1256.2	19.05	20.10	1178	1306.9	114.33	24.94	1122	1151.9	18.58	**10.12**	1153	1181.0	15.92	12.91
la22	927	1091	1134.2	20.45	22.35	1053	1186.1	124.60	27.95	989	1032.5	25.33	11.38	1008	1028.0	8.65	**10.90**
la23	1032	1141	1185.5	20.63	14.87	1093	1167.3	58.25	13.11	1035	1085.7	30.06	5.20	1069	1082.1	7.26	**4.85**
la24	935	1104	1138.4	16.32	21.75	1053	1166.7	83.63	24.78	1009	1036.5	15.88	10.86	991	1028.8	14.32	**10.03**
la25	977	1148	1183.8	15.45	21.17	1101	1206.4	85.01	23.48	1052	1108.3	31.43	13.44	1084	1102.0	9.89	**12.79**
la26	1218	1440	1459.5	13.49	19.83	1393	1468.6	70.70	20.57	1298	1346.9	25.67	**10.58**	1330	1366.2	16.73	12.17
la27	1235	1461	1511.8	21.91	22.41	1387	1480.7	69.30	19.89	1336	1368.9	19.98	**10.84**	1415	1428.0	7.17	15.63
la28	1216	1443	1474.3	19.04	21.24	1388	1446.7	67.34	18.97	1284	1362.0	42.11	**12.01**	1372	1388.2	8.53	14.16
la29	1152	1432	1460.0	13.39	26.74	1415	1478.6	52.67	28.35	1330	1371.1	34.09	19.02	1353	1367.1	12.57	**18.67**
la30	1355	1517	1567.8	29.13	15.70	1512	1625.1	77.96	19.93	1437	1489.9	33.14	9.96	1443	1467.6	14.35	**8.31**
orb01	1059	1229	1267.6	22.18	19.70	1183	1304.2	50.10	23.15	1142	1183.5	23.17	11.76	1147	1157.9	10.45	**9.34**
orb02	888	999	1021.9	17.25	15.08	1018	1086.3	56.61	22.33	932	948.1	15.44	**6.77**	954	966.7	10.41	8.86
orb03	1005	1223	1253.7	16.12	24.75	1286	1367.5	65.42	36.07	1064	1144.0	41.62	13.83	1108	1132.6	13.65	**12.70**
orb04	1005	1111	1157.6	23.69	15.18	1062	1172.6	57.89	16.68	1023	1104.6	48.08	9.91	1058	1093.7	16.95	**8.83**
orb05	887	1023	1064.8	21.43	20.05	1014	1112.6	56.04	25.43	914	953.6	25.03	7.51	915	937.0	11.78	**5.64**
orb06	1010	1216	1231.3	8.60	21.91	1271	1316.8	49.03	30.38	1063	1130.7	41.10	11.95	1106	1125.0	14.16	**11.39**
orb07	397	455	464.9	6.82	17.10	435	476.1	25.94	19.92	412	423.8	7.95	**6.75**	411	425.7	6.94	7.23
orb08	899	1059	1091.2	17.09	21.38	1120	1176.6	36.46	30.88	956	998.8	30.46	**11.10**	993	1022.1	11.68	13.69
orb09	934	1052	1076.6	18.27	15.27	1030	1152.5	70.89	23.39	1001	1042.5	48.46	11.62	993	1017.8	11.57	**8.97**
orb10	944	1103	1136.5	20.71	20.39	1053	1209.2	115.39	28.09	966	1031.3	34.18	**9.25**	1023	1038.9	13.22	10.05
+/ = /–			0/8/40				0/2/46				16/7/25				–		

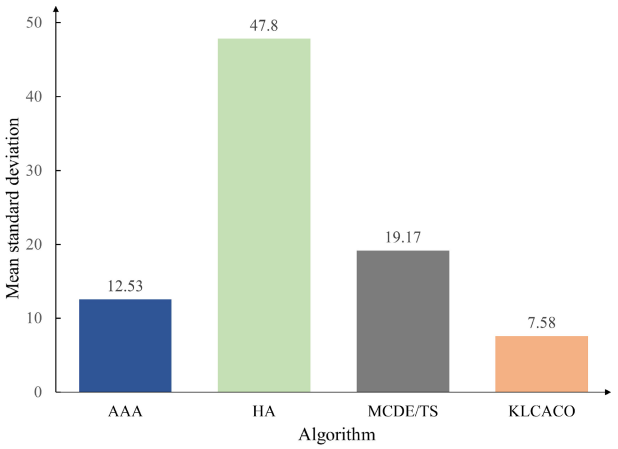

Fig. 8 Mean standard deviation comparison of the algorithms.

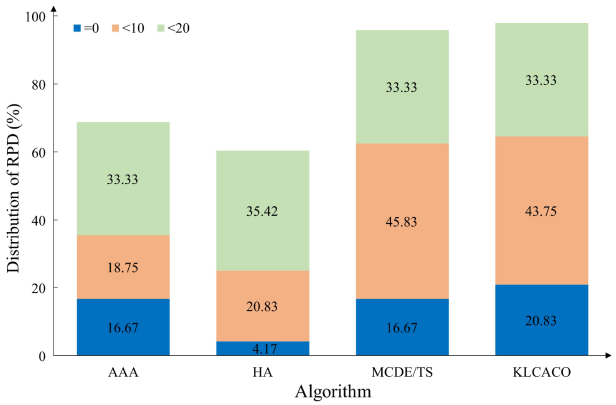

Fig. 9 Relative percentage error distribution of algorithms.

maximum evaluation time of the algorithm. Therefore, a hierarchical comparison of the RPD is conducted, and the proportion of the solving effectiveness of the algorithm is calculated based on the criteria "RPD = 0, RPD < 10, RPD < 20" to generate Fig. 9. This figure shows that the KLCACO algorithm outperforms other algorithms at the "RPD = 0" level and almost all results obtained using the KLCACO algorithm fall within the "RPD < 20" range. Additionally, the algorithm proposed in this paper displays superior results compared to the AAA and HA algorithms. Therefore, the KLCACO algorithm can provide competitive performance for solving JSPs.

Under the constraint of limited evaluation iterations, the KLCACO algorithm demonstrated excellent performance on most of the datasets. This finding effectively validates the efficiency of the KLCACO algorithm in solving JSP. Furthermore, in terms of algorithm stability and robustness, the KLCACO algorithm displayed commendable results. These results indicate that although optimal solutions have not been attained for all test cases, the proposed algorithm maintains a high level of competitiveness due to its robustness and stability.

5 Conclusion

This paper introduces a co-operative-guided ACO algorithm with knowledge learning for efficiently addressing JSP. The primary improvements introduced in the algorithm are as follows:

For the characteristics of the JSP, the KLCACO algorithm uses cooperative construction to achieve multimachine synchronous processing operation selection and proposes an individual solution construction scheme based on schedule knowledge learning. The pheromone structure optimization is completed by incorporating the parallel processing structure of the machine, and the JSP solution information is used for adjusting the pheromone threshold and estimating the population evolution state to complete the dynamic adaptive control of the pheromone. Additionally, the N7 neighborhood is used in the local information optimization of the KLCACO algorithm, further improving the local mining capability and enhancing the efficiency of neighborhood search. Numerous experiments were conducted to verify the performance of the KLCACO algorithm, and the results were comprehensively analyzed. The results show that the KLCACO algorithm has good stability and high solution accuracy. Compared with other high-quality intelligent optimization algorithms, the KLCACO algorithm exhibits excellent performance.

Moreover, the KLCACO algorithm occasionally fails to achieve the optimal solution within the limited time, indicating potential avenues for enhancing existing knowledge model mechanisms. Subsequent endeavors in future research will focus on refining this issue to facilitate optimal information learning and achieve improved performance.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Nos. 62366003 and 62066019), the Natural Science Foundation of Jiangxi Province (No. 20232BAB202046), and the Graduate Innovation Foundation of Jiangxi University of Science and Technology (No. XY2022-S040).

References

[1] S. M. Johnson, Optimal two- and three-stage production schedules with setup times included, *Nav. Res. Logist. Q.*, vol. 1, no. 1, pp. 61–68, 1954.

[2] M. R. Garey, D. S. Johnson, and R. Sethi, The complexity of flowshop and jobshop scheduling, *Math. Oper. Res.*, vol. 1, no. 2, pp. 117–129, 1976.

[3] Y. Fu, Y. Hou, Z. Wang, X. Wu, K. Gao, and L. Wang, Distributed scheduling problems in intelligent manufacturing systems, *Tsinghua Science and Technology*, vol. 26, no. 5, pp. 625–645, 2021.

[4] C. Destouet, H. Tlahig, B. Bettayeb, and B. Mazari, Flexible job shop scheduling problem under Industry 5.0: A survey on human reintegration, environmental consideration and resilience improvement, *J. Manuf. Syst.*, vol. 67, pp. 155–173, 2023.

[5] F. Zhao, Z. Xu, L. Wang, N. Zhu, T. Xu, and J. Jonrinaldi, A population-based iterated greedy algorithm for distributed assembly No-wait flow-shop scheduling problem, *IEEE Trans. Ind. Inform.*, vol. 19, no. 5, pp. 6692–6705, 2023.

[6] G. G. Wang, D. Gao, and W. Pedrycz, Solving multiobjective fuzzy job-shop scheduling problem by a hybrid adaptive differential evolution algorithm, *IEEE Trans. Ind. Inform.*, vol. 18, no. 12, pp. 8519–8528, 2022.

[7] B. Yan, M. A. Bragin, and P. B. Luh, An innovative formulation tightening approach for job-shop scheduling, *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 3, pp. 2526–2539, 2022.

[8] Y. J. Yao, Q. H. Liu, X. Y. Li, and L. Gao, A novel MILP model for job shop scheduling problem with mobile robots, *Robot. Comput. Integr. Manuf.*, vol. 81, p. 102506, 2023.

[9] M. A. Şahman and S. Korkmaz, Discrete artificial algae algorithm for solving job-shop scheduling problems, *Knowl. Based Syst.*, vol. 256, p. 109711, 2022.

[10] S. Mahmud, A. Abbasi, R. K. Chakrabortty, and M. J. Ryan, Multi-operator communication based differential evolution with sequential Tabu Search approach for job shop scheduling problems, *Appl. Soft Comput.*, vol. 108, p. 107470, 2021.

[11] J. Xie, X. Li, L. Gao, and L. Gui, A hybrid algorithm with a new neighborhood structure for job shop scheduling problems, *Comput. Ind. Eng.*, vol. 169, p. 108205, 2022.

[12] L. N. Xing, Y. W. Chen, P. Wang, Q. S. Zhao, and J. Xiong, A knowledge-based ant colony optimization for flexible job shop scheduling problems, *Appl. Soft Comput.*, vol. 10, no. 3, pp. 888–896, 2010.

[13] W. Zhang, W. Hou, C. Li, W. Yang, and M. Gen, Multidirection update-based multiobjective particle swarm optimization for mixed No-idle flow-shop scheduling problem, *Complex Syst. Model. Simul.*, vol. 1, no. 3, pp. 176–197, 2021.

[14] J. Li, X. Gu, Y. Zhang, and X. Zhou, Distributed flexible job-shop scheduling problem based on hybrid chemical reaction optimization algorithm, *Complex Syst. Model. Simul.*, vol. 2, no. 2, pp. 156–173, 2022.

[15] X. Han, Y. Han, Q. Chen, J. Li, H. Sang, Y. Liu, Q. Pan, and Y. Nojima, Distributed flow shop scheduling with sequence-dependent setup times using an improved iterated greedy algorithm, *Complex Syst. Model. Simul.*, vol. 1, no. 3, pp. 198–217, 2021.

[16] X. Zhang, B. Zhang, L. Meng, Y. Ren, R. Meng, and J. Li, An evolutionary algorithm for a hybrid flowshop scheduling problem with consistent sublots, *Int. J. Autom. Contr.*, vol. 16, no. 1, p. 19, 2022.

[17] L. Wang, Z. Pan, and J. Wang, A review of reinforcement learning based intelligent optimization for manufacturing scheduling, *Complex Syst. Model. Simul.*, vol. 1, no. 4, pp. 257–270, 2021.

[18] B. Xi and D. Lei, Q-learning-based teaching-learning optimization for distributed two-stage hybrid flow shop scheduling with fuzzy processing time, *Complex Syst. Model. Simul.*, vol. 2, no. 2, pp. 113–129, 2022.

[19] F. Zhao, T. Jiang, and L. Wang, A reinforcement learning driven cooperative meta-heuristic algorithm for energy-efficient distributed No-wait flow-shop scheduling with sequence-dependent setup time, *IEEE Trans. Ind. Inform.*, vol. 19, no. 7, pp. 8427–8440, 2023.

[20] X. Wu, Z. Cao, and S. Wu, Real-time hybrid flow shop scheduling approach in smart manufacturing environment, *Complex Syst. Model. Simul.*, vol. 1, no. 4, pp. 335–350, 2021.

[21] J. J. Wang and L. Wang, A cooperative memetic algorithm with feedback for the energy-aware distributed flow-shops with flexible assembly scheduling, *Comput. Ind. Eng.*, vol. 168, p. 108126, 2022.

[22] Z. Pan, D. Lei, and L. Wang, A Bi-population evolutionary algorithm with feedback for energy-efficient fuzzy flexible job shop scheduling, *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 52, no. 8, pp. 5295–5307, 2022.

[23] Z. Chen, L. Zhang, X. Wang, and P. Gu, Optimal design of flexible job shop scheduling under resource preemption based on deep reinforcement learning, *Complex Syst. Model. Simul.*, vol. 2, no. 2, pp. 174–185, 2022.

[24] C. Zhang, P. Li, Z. Guan, and Y. Rao, A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem, *Comput. Oper. Res.*, vol. 34, no. 11, pp. 3229–3242, 2007.

[25] M. Dorigo and T. Stützle, *Ant colony optimization*: *Overview and recent advances*. M. Gendreau and J. Y. Potvin, eds. Cham, Switzerland: Springer, 2019, pp. 311–351.

[26] M. Dorigo, V. Maniezzo, and A. Colorni, Ant system: Optimization by a colony of cooperating agents, *IEEE Trans. Syst. Man Cybern. Part B Cybern.*, vol. 26, no. 1, pp. 29–41, 1996.

[27] M. Dorigo and L. M. Gambardella, Ant colony system: A cooperative learning approach to the traveling salesman problem, *IEEE Trans. EComput.*, vol. 1, no. 1, pp. 53–66, 1997.

[28] T. Stutzle and H. Hoos, MAX-MIN ant system and local search for the traveling salesman problem, in *Proc. 1997 IEEE Int. Conf. Evolutionary Computation*, Indianapolis, IN, USA, 1997, pp. 309–314.

[29] P. Korytkowski, S. Rymaszewski, and T. Wiśniewski, Ant

colony optimization for job shop scheduling using multi-attribute dispatching rules, *Int. J. Adv. Manuf. Technol.*, vol. 67, no. 1, pp. 231–241, 2013.
[30] E. Florez, W. Gomez, and M. Lola Bautista, An ant colony optimization algorithm for job shop scheduling problem, *Int. J. Artif. Intell. Appl.*, vol. 4, no. 4, pp. 53–66, 2013.
[31] E. Falkenauer and S. Bouffouix, A genetic algorithm for job shop, in *Proc. 1991 IEEE Int. Conf. Robotics and Automation*, Sacramento, CA, USA, 2002, pp. 824–829.
[32] K. Sun, D. Zheng, H. Song, Z. Cheng, X. Lang, W. Yuan, and J. Wang, Hybrid genetic algorithm with variable neighborhood search for flexible job shop scheduling problem in a machining system, *Expert Syst. Appl.*, vol. 215, p. 119359, 2023.
[33] M. L. Pinedo, Job shops (deterministic), in *Scheduling*, Cham, Switzerland: Springer, 2016. pp. 183–220,
[34] W. Zhang, X. Chen, and J. Jiang, A multi-objective optimization method of initial virtual machine fault-tolerant placement for star topological data centers of cloud systems, *Tsinghua Science and Technology*, vol. 26, no. 1, pp. 95–111, 2021.
[35] J. Liu, S. Anavatti, M. Garratt, and H. A. Abbass, Modified continuous ant colony optimisation for multiple unmanned ground vehicle path planning, *Expert Syst. Appl.*, vol. 196, p. 116605, 2022.
[36] W. Li, C. Wang, Y. Huang, and Y. M. Cheung, Heuristic smoothing ant colony optimization with differential information for the traveling salesman problem, *Appl. Soft Comput.*, vol. 133, p. 109943, 2023.
[37] Y. H. Jia, Y. Mei, and M. Zhang, A bilevel ant colony optimization algorithm for capacitated electric vehicle routing problem, *IEEE Trans. Cybern.*, vol. 52, no. 10, pp. 10855–10868, 2022.
[38] T. Stützle and H. H. Hoos, Max-min ant system, *Future Gener. Comput. Syst.*, vol. 16, no. 8, pp. 889–914, 2000.
[39] J. Adams, E. Balas, and D. Zawack, The shifting bottleneck procedure for job shop scheduling, *Manag. Sci.*, vol. 34, no. 3, pp. 391–401, 1988.
[40] H. Fisher, Probabilistic learning combinations of local jobshopscheduling rules, Industrial scheduling, https://cir.nii.ac.jp/crid/1573950398887935360, 2023.
[41] S. Lawrence, Resouce constrained project scheduling: Anexperimental investigation of heuristic scheduling-techniques (supplement), http://refhub.elsevier.com/S0950-7051(22)00866-8/sb73, 2023.
[42] D. Applegate and W. Cook, A computational study of the job-shop scheduling problem, *ORSA J. Comput.*, vol. 3, no. 2, pp. 149–156, 1991.
[43] J. Yu, X. You, and S. Liu, A heterogeneous guided ant colony algorithm based on space explosion and long–short memory, *Appl. Soft Comput.*, vol. 113, p. 107991, 2021.

Xiangfang Yan received the BEng degree in computer science and technology from Jiangxi University of Science and Technology, China in 2021. He is currently a master student at School of Information Engineering, Jiangxi University of Science and Technology, China. His main research interests are combinatorial optimization problems and ant colony optimization algorithms.

Ying Huang received the MEng degree in computer application technology from Jiangxi University of Science and Technology, China in 2008, and the PhD degree in computer software and theory from Wuhan University, China in 2016. Currently, she works at Gannan Normal University as an associate professor. She has published over 20 research papers in international conferences and journals. Her research interests include the areas of service computing, business processes, and intelligent optimization algorithms.

Wei Li received the BEng degree in computer science and technology from Jiangxi University of Science and Technology, China in 2003, the MEng degree in computer application technology from Jiangxi University of Science and Technology, China in 2008, and the PhD degree from South China Agricultural University, China in 2018. He is currently an associate professor at School of Information Engineering, Jiangxi University of Science and Technology, China. He has over 30 papers published in fully refereed international journals and conferences, and served as the program chair or program committee member in many international conferences. His research focuses mainly on evolutionary computation, large-scale optimization, evolving deep neural networks, and multiobjective machine learning.