Hindawi
Scientific Programming
Volume 2017, Article ID 9016303, 11 pages
https://doi.org/10.1155/2017/9016303

Research Article

Flexible Job Shop Scheduling Problem Using an Improved Ant Colony Optimization

Lei Wang, Jingcao Cai, Ming Li, and Zhihu Liu

School of Mechanical and Automotive Engineering, Anhui Polytechnic University, Wuhu 241000, China

Correspondence should be addressed to Lei Wang; wangdalei2000@126.com

Received 27 May 2016; Revised 31 October 2016; Accepted 24 November 2016; Published 26 January 2017

Academic Editor: Fabrizio Riguzzi

As an extension of the classical job shop scheduling problem, the flexible job shop scheduling problem (FJSP) plays an important role in real production systems. In FJSP, an operation is allowed to be processed on more than one alternative machine. It has been proven to be a strongly NP-hard problem. Ant colony optimization (ACO) has been proven to be an efficient approach for dealing with FJSP. However, the basic ACO has two main disadvantages including low computational efficiency and local optimum. In order to overcome these two disadvantages, an improved ant colony optimization (IACO) is proposed to optimize the makespan for FJSP. The following aspects are done on our improved ant colony optimization algorithm: select machine rule problems, initialize uniform distributed mechanism for ants, change pheromone's guiding mechanism, select node method, and update pheromone's mechanism. An actual production instance and two sets of well-known benchmark instances are tested and comparisons with some other approaches verify the effectiveness of the proposed IACO. The results reveal that our proposed IACO can provide better solution in a reasonable computational time.

1. Introduction

Scheduling problem plays a very important role in many industrial systems [1]. Therefore it has attracted considerable researches for recent decades [2–7]. Job shop scheduling problem (JSP) is a branch of production scheduling and combinatorial optimization problems [8]. The flexible job shop scheduling problem (FJSP) is an extension of the job shop scheduling problem (JSP) [9]. Different from JSP, an operation can be processed on more than one candidate machines in FJSP. As a result, two subproblems facing FJSP are machine assignment and operation sequencing. Machine assignment is how to assign a machine for each operation while operation sequencing is how to schedule all operations on machines to optimize the given performance indicators [10]. Thus, FJSP is more complicated than the classical JSP and it has been proven to be a strongly NP-hard in 1993 [11].

The FJSP was first studied by Brucker and Schlie who used a polynomial approach to deal with two jobs FJSP [12]. In recent years, a large number of heuristics or metaheuristics have been employed to deal with FJSP, specifically through tabu search (TS) [13], simulated annealing (SA) [14], genetic algorithm (GA) [15, 16], particle swarm optimization (PSO) [17, 18], ant colony optimization (ACO) [19], artificial bee colony (ABC) [20], and hybrid approaches based on different heuristics and metaheuristics.

Among these metaheuristics, ACO has been proved to be an efficient approach for dealing with JSPs [21–23]. However, this approach still has some limitations in practice: (1) a lot of computational time will be spent on obtaining the ideal solution. (2) The search usually falls into local optimal solution. Therefore, in order to overcome these limitations, numerous improved ACO algorithms or hybrid ACO algorithms have been developed. A knowledge-based ACO approach for solving flexible job shop scheduling problems is proposed in [19]. An improved ACO was proposed to deal with dynamic hybrid flow shop scheduling in [24]. A modified ACO called two-pheromone ant colony optimization was proposed for solving flexible job shop scheduling problem with due window in [25]. Ant colony optimization combined with tabu search was employed to deal with JSP in [26]. Two-generation Pareto ant colony algorithm was proposed by Zhao et al. for solving multiobjective job shop scheduling problem [27]. A two-stage ant colony optimization was presented to minimize

the makespan in [28]. Leung et al. [29] proposed an agent-based ant colony optimization for solving integrated process planning and scheduling problem.

Great efforts have been done for solving job shop scheduling problems or flexible job shop scheduling problems by improving ACO algorithms; however, these improved ACO algorithms are almost achieved by changing pheromone update mechanism. Although this can improve the search speed and solution efficiency, excessively strengthening the pheromone feedback of the best path may easily lead to premature convergence. Therefore, in order to solve these problems that exist in the basic ACO or improved ACO algorithms mentioned above, we propose an improved ant colony optimization (IACO) to solve the FJSP in this paper and the results are found to be closer or equal to the global optimum.

The remainder of this paper is organized as follows. Section 2 describes the model formulation for FJSP. The proposed IACO is introduced in Section 3. Experimental test, comparison, and discussion are reported in Section 4. The conclusions and future work are given in Section 5.

2. Description of FJSP

The $n \times m$ FJSP can be depicted as follows [30]: there are n jobs and m machines. Each job i comprises n_i operations $\{o_{i1}, o_{i2}, o_{i3}, \ldots, o_{in_i}\}$. Each operation o_{ik} can be processed by only one machine from the candidate machine set A_{ik}. The assumptions for FJSP are as follows:

(1) Each machine can be used at time zero.

(2) Each job can be processed at time zero.

(3) Each machine can process only one operation at a time.

(4) Once an operation starts on a machine, it cannot be interrupted.

(5) The sequences of operations for all jobs are prespecified.

(6) Neither due dates nor release times are specified.

(7) The transportation times among machines are not considered.

(8) All machines are not always identical.

The objective is to minimize the makespan and the mathematical model of the FJSP is shown as follows [31]:

$$\min \quad f(x) = C_M = \max_{1 \le i \le n} \left\{ c_{in_i} \right\}, \tag{1}$$

$$\text{s.t.} \quad \left[\left(c_{hg} - c_{ik} - t_{hgj} \right) \cdot x_{hgj} \ge 0 \vee \left(c_{ik} - c_{hg} - t_{ikj} \right) \cdot x_{ikj} \ge 0 \right], \quad \forall i, j, g.h, \tag{2}$$

$$c_{in_i} - c_{i(k-1)} \ge t_{ikj} \cdot x_{ikj}, \quad k = 2, \ldots, n_i, \ \forall i, j, \tag{3}$$

$$\sum_{x_{ikj} \in A_{ik}} x_{ikj} = 1, \quad \forall i, j, k, \tag{4}$$

$$x_{ikj} \in \{0, 1\}, \quad \forall i, j, k, \tag{5}$$

$$c_{ik} \ge 0, \quad \forall i, k, \tag{6}$$

where i, h is the job index, $i, h = 1, 2, \ldots, n$. j is the machine index, $j = 1, 2, \ldots, m$. k, g is the operation index, $k, g = 1, 2, \ldots, n_i$. n represents total jobs. m represents total machines. n_i represents total operations of job i. t_{ikj} represents processing time of kth operation of job i on machine j. c_{ik} is the completed time of o_{ik}. $x_{ikj} = 1$, if machine j is selected for o_{ik}; otherwise, $x_{ikj} = 0$.

Equations (2) and (3) are the operation sequence constraint. Equation (4) indicates that each operation can only be processed on one machine from machine set at a time. Equations (5) and (6) are decision variables which are 0-1 binary variable and nonnegative, respectively.

3. Improved Ant Colony Optimization for FJSP

3.1. Principles of Ant Colony Optimization. As social insects, ants live in colonies and their behavior is governed by the goal of colony survival rather than being focused on the survival of individuals. The main idea of the ACO is inspired by the behavior of real ants searching for food. The real ants can communicate with each other about food sources through pheromone. When the real ants move along, they release pheromone on the path they have passed. Other ants are attracted to follow them by observing the pheromone trail.

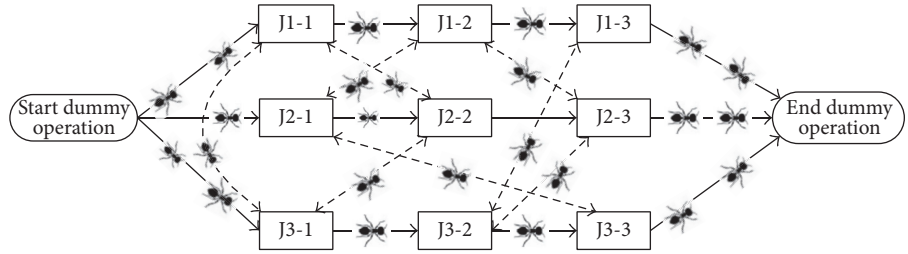

Figure 1: Ants' moving disjunctive graph.

Therefore, the path is enhanced and thus attracts more ants [32, 33]. Compared with some other heuristics, the ACO is characterized by distributed computation and positive feedback [34].

If the operations are looked at as ants, it can be seen easily that there exist lots of similarities between an ant colony's foraging process and FJSP. Operations must search for proper machine to process them. Like ants, they want to search for the shortest path. Ants' nest and food source are similar with start and end dummy operation, respectively. If we look at an operation as an ant's path for foraging food, then the relation of any two operations can be looked at as an alternative path, and different processing time of all the operations on the machine just like different length of paths.

The disjunctive graph of an FJSP example can be described in Figure 1. There are three jobs and each job has three operations. J1-1, J1-2, and J1-3 represent three operations of job 1. J2-1, J2-2, and J2-3 represent three operations of job 2. J3-1, J3-2, and J3-3 represent three operations of job 3. Each job must comply with the process sequence constraints. According to the constraints of process sequence and machine occupancy, ants travel total nine operations of the three jobs and search for the operation order on each machine and then gain the optimal or near-optimal solution for the flexible job shop scheduling problem. Therefore it is feasible to use ACO to solve the flexible job shop scheduling problems.

The key steps of the basic ACO are the calculations of transition probability, visibility, and pheromone amount. The nodes or goal point are chosen by ants according to pheromone amount and visibility [35]. At time t, the probability for ant k choosing the path from point i to j is calculated as follows:

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{\tau_{ij}^{\alpha}(t)\,\eta_{ij}^{\beta}(t)}{\sum_{s\in \text{allowed}_k} \tau_{is}^{\alpha}(t)\,\eta_{is}^{\beta}(t)}, & \text{if } j \in \text{allowed}_k \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

where the allowed_k is selectable machines group that ant k can choose and α and β are the pheromone and expectation factor, respectively. $\eta_{ij}(t)$ is heuristic factor and it is computed as $\eta_{ij}(t) = 1/f(x)$. The more pheromone value on the path is and the higher visibility is and the bigger probability for choosing this path is.

As time goes on, the pheromone of the path evaporates gradually. An ant will modify the pheromone value on the passed edges by applying the local updating rule, as follows:

$$\tau_{ij}(t) = (1-\rho)\cdot\tau_{ij}(t) + \rho\tau_0, \tag{8}$$

where ρ is the evaporation coefficient of pheromone and $0 < \rho < 1$. τ_0 is the initial value of pheromone on each path.

Once all ants have arrived at their destination, the value of pheromone on the edge is modified again by applying the global updating rule, as follows:

$$\tau_{ij}(t) = (1-\rho)\cdot\tau_{ij}(t) + \rho\Delta\tau_{ij}^{k}(t), \tag{9}$$

where $\Delta\tau_{ij}^{k}(t)$ is the quantity of pheromone on the path (i, j) laid by ant k, and it can be defined as follows:

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \dfrac{Q}{f(x)^{k}} & \text{if ant } k \text{ travel through arc } (i,j) \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

In (10), Q is a constant and it denotes the strength of pheromone; $f(x)^{k}$ is evaluation value of the kth ant after finishing the search task.

As mentioned above, the basic ACO still has some weaknesses in practice. For example, the search usually gets trapped in local optimal solution. Meanwhile it needs a lot of computational time to obtain the ideal solution. Therefore, we propose an improved ant colony optimization to solve the FJSP in this paper.

3.2. Improved Aspects on Ant Colony Optimization Algorithm. The following improvements are done for machine selection: the machine which has the shortest processing time for finishing the jth operation of job i is selected by using 60% probability; the machine which has the shortest time for processing the jth operation of job i is selected by using 30% probability; the machine is randomly selected by using 10% probability, and the random selection can be achieved by using roulette selection method. The range for selecting machine can be expanded by using these three kinds of selection method. Take Figure 2, for instance, $T_1 < T_2 < T_3$; therefore M_1 is selected with a 60% probability, M_2 is selected with a 30% probability, and one machine among M_1, M_2, and M_3 is randomly selected with a 10% probability.

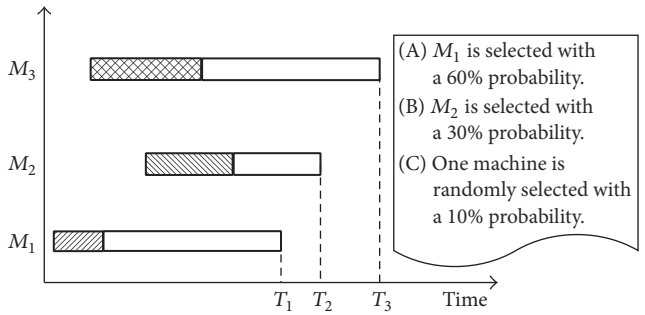

Figure 2: Gantt chart for machine selection.

The main steps of the basic ACO are the initialization position of ants, the calculation of transition probability, visibility, and pheromone value, as mentioned in Section 3.1.

When some other control parameters remain unchangeable, the initial position of the ants has a greater influence on the ACO. The ants should be distributed uniformly on the set which contains the first operations of all jobs, and the probability for searching for the global ideal solution will become much greater. If a large number of ants search for food from the same starting point, the solution diversity may be lost. Therefore, an initialization mechanism is used to distribute uniformly the ants' initial positions.

In the initial search stage, some paths are passed by ants, and some other paths are not passed. If an ant searches for path according to the pheromone's guiding mechanism, it is easy to reduce the probability for selecting the path that has not been passed yet, and therefore the chance for ants to find the global ideal solution will be reduced. So when the pheromone exceeds a certain value, the ants are allowed to find the optimal path according to pheromone's guiding mechanism, as follows:

$$\tau_{ij}(t) \geq 1.1 * \tau_0, \tag{11}$$

where τ_0 is the initial value of pheromone on each path and $\tau_{ij}(t)$ is pheromone value between node i and j (path $i \rightarrow j$) at time t.

In order to expand the search scope of the ants and improve the search space of the basic ACO, initializing pheromone needs to be done when the pheromone value on a path is more than 90% of the total pheromone value on all paths (shown in (12)) because the ACO has fallen into local optimum in this case.

$$\tau_{ij}(t) \geq 0.9 * \tau_{\text{Sum}}(t), \tag{12}$$

where $\tau_{\text{Sum}}(t)$ is the total pheromone value on all paths.

However, the next available path selected by employing the transition probability does not always obtain the optimal direction for the basic ACO, and the pheromone deviated from the ideal solution has the potential to be enhanced, which will easily lead to the local optimal solution.

After the transition probability for each candidate is obtained, if the roulette selection method is adopted, not only the path with a large transition probability is likely to be selected, but also the path with a small transition probability has the opportunity to be selected; thereby the search space and solution quality can be expanded and improved, respectively.

Therefore, a new node selection method, combining prior knowledge, probability search, and random search, is proposed in this paper. When the search is trapped in local optimal solution, the solution space can be further searched by adjusting the pheromone and increasing the random selection probabilities.

$$p = \begin{cases} p_1 = \arg\max\limits_{j \in \text{allowed}_k} \left\{ \left[\tau_{ij}(t)\right]^{\alpha} \cdot \left[\eta_{ij}(t)\right]^{\beta} \cdot \xi_{ij}(t) \right\} & 0 \leq q \leq q_0 \\ p_2 = p_{ij}^{k}(t) = \begin{cases} \dfrac{\tau_{ij}^{\alpha}(t)\,\eta_{ij}^{\beta}(t)\,\xi_{ij}(t)}{\sum_{s \in \text{allowed}_k} \tau_{is}^{\alpha}(t)\,\eta_{is}^{\beta}(t)\,\xi_{ij}(t)}, & j \in \text{allowed}_k \\ 0 & \text{otherwise} \end{cases} & q_0 < q \leq q_1 \\ p_3 = \text{random search} & q_1 < q \leq 1, \end{cases} \tag{13}$$

where q is a random value and $0 \leq q < 1$; q_0 is the degree of prior knowledge; q_1 is the lower bound level of random search; $\xi_{ij}(t)$ represents the appearing number of arc (i, j) during previous iterations for searching good solution.

The more the arc (i, j) appears, the more the role is played for searching good solution by using positive feedback.

A good balance relation between "using the past information to speed up the convergence" and "exploring new paths" can be established by combining these three selection methods. By this way, the search space can be enlarged and the global good solution can be obtained with a larger probability.

In addition, a concept of the invalid search number is defined, which means the difference between the current number of iterations N_1 and the recent number of iterations N_2 for improving the solution, as shown in (14). When the number of the invalid search exceeds a specified value N_0, the algorithm is considered as a local optimum, as shown in (15).

$$N = N_1 - N_2, \tag{14}$$

$$N \geq N_0. \tag{15}$$

Meanwhile, the maximum or minimum pheromone trails may lead to premature convergence for searching solution.

Therefore, the maximal pheromone trail $\tau_{\max}$ and the minimal pheromone trail $\tau_{\min}$ are given in our IACO in order to make all pheromone trails $\tau_{ij}(t)$ satisfy $\tau_{\min} \leq \tau_{ij}(t) \leq \tau_{\max}$. This idea is inspired by the Max–Min ant system [36].

When the number of the invalid searches exceeds a specified value N_0, the pheromone on the path is forcedly destroyed in order to avoid falling into the local optimum. In this paper, the value is reduced to sixty percent of the original pheromone $\tau_{ij}(t)$, as follows:

$$\tau_{ij}(t+n) = \begin{cases} 60\% * \tau_{ij}(t) & \text{if } N \geq N_0 \\ \text{Eq. (9)} & \text{otherwise.} \end{cases} \quad (16)$$

3.3. Steps of the IACO. The specific implementation steps of the proposed IACO for solving FJSP are shown as follows.

Step 1. Initialize parameters α, β, ρ, Q, q_0, q_1, and τ_0 and tabu list.

Step 2. Initialize the ants' initial positions by using uniform distribution mechanism.

Step 3. Select machine by using our proposed machine selection strategy.

Step 4. Establish three sets: one set S_1 contains operations that have been already visited by ants. One set S_2 contains operations for next candidates. Another set S_3 contains operations that are waiting to be added to the candidate set. And add the first operation of each job to the set of next candidate operation waiting to be selected.

Step 5. Judge whether the pheromone $\tau_{ij}(t)$ on the path is greater or equal to $1.1 * \tau_0$ or not. If not, then select next operation in a random manner. If so, then select the next path according to (13). After the transition probability for each candidate is obtained, the roulette selection method is adopted to select the next path or node.

Step 6. Add the just selected operation to set S_1 and remove the just selected operation from set S_1. Meanwhile, add the next operation to set S_2 and remove the next operation from set S_3.

Step 7. Judge whether there are subsequent operations or not. If there are subsequent operations, then go to Step 5. Otherwise go to Step 8.

Step 8. Calculate the good solution of this iteration and save.

Step 9. Find the best ant and update its global pheromone.

Step 10. Calculate the pheromone value on each path and limit the pheromone value on each path within the range of two values $\tau_{\max}$ and $\tau_{\min}$. Meanwhile, judge whether $\tau_{ij}(t)$ is not less than $0.9 * \tau_{\text{Sum}}(t)$. If so, initialize the pheromone value of that path.

Step 11. Judge whether N is more than N_0 or not. If it is satisfied, then forcedly destroy the pheromone value of that path according to (16).

Step 12. End the algorithm either when the optimal or near-optimal solution is found or when a maximal iteration is satisfied and then output the global optimal or near-optimal solution. Otherwise go to Step 2.

The framework of our IACO is shown in Figure 3.

4. Experimental Results

To test the performance of the proposed IACO, two groups of simulation experiments are executed. One group of experiments comes from an actual production instance, and the other group of experiments comes from benchmark problems.

4.1. Test on the Actual Production Instance. Table 1 shows the process information from an actual production instance. The value in Table 1 represents processing time on each machine and "—" means the operation cannot be processed on that machine.

The parameters and their values, which are used for running our IACO, are shown as follows: number of ants $m = 39$, weight of pheromone trail $\alpha = 1$, weight of heuristic information $\beta = 2$, pheromone evaporation parameter $\rho = 0.1$, the initial value of pheromone value $\tau_0 = 0.1$, constant for pheromone updating $Q = 120$, premature constant $N_0 = 20$, $\tau_{\max} = 10$, $\tau_{\min} = 0.01$, the value of prior knowledge $q_0 = 0.3$, the lower bound value of random search $q_1 = 0.8$, and number of iterations $G = 100$.

The Gantt charts obtained by our IACO, the basic ACO, and Max–Min Ant System (MMAS) are shown in Figures 4, 5, and 6. In addition, Figure 7 depicts the convergence curves of the best makespan by using our IACO, the basic ACO, and MMAS. To such an instance, from Figure 7, it can be seen that our IACO obtains the ideal makespan (25) with 8 iterations. The basic ACO only obtains the ideal makespan (28) and it needs 28 iterations. Though the MMAS can obtain the ideal makespan (25), however, it needs as many as 61 iterations. Therefore, our proposed IACO is very efficient for solving FJSP.

In order to prove the solving solution quality and solving efficiency of our IACO, Table 2 shows the experimental results of the proposed IACO in comparison to the basic ACO and MMAS by 10 independent experiments. The best solution, the average solution, the worst solution, and the average computing time of our proposed IACO are the best among the compared methods. The results demonstrate that the proposed IACO is effective and efficient in solving FJSP.

4.2. Test on Kacem Benchmark Instances. FJSP was classified into two categories by Kacem et al. [37]: (1) Partial FJSP (P-FJSP) which means that each operation can be processed by a subset of machines and (2) Total FJSP (T-FJSP) which means that each operation can be processed by all machines.

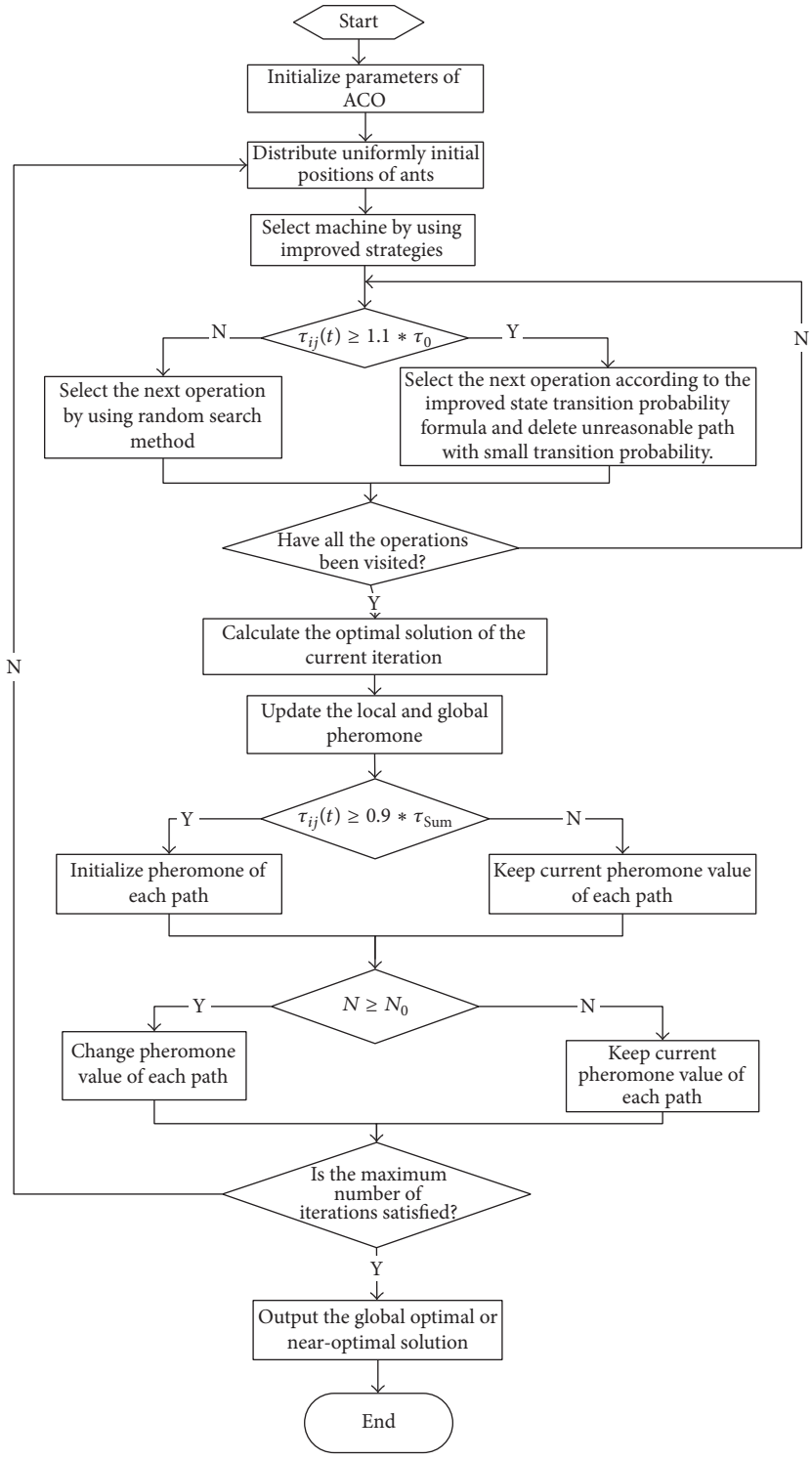

Figure 3: The flowchart of the proposed IACO.

Firstly, one 8 × 8 P-FJSP and one 10 × 10 T-FJSP from benchmark instances are applied to evaluate the performance of our proposed IACO.

The Gantt chart and the convergence curve for 8 × 8 benchmark P-FJSP obtained by our IACO are shown in Figures 8 and 9, respectively. The Gantt chart and the convergence curve for 10 × 10 benchmark T-FJSP obtained by our IACO are shown in Figures 10 and 11, respectively. To such an instance, from Figure 8–11, it can be seen that our IACO can easily obtain the best makespan with very few iterations.

In addition, we compare the performances of our proposed IACO with several methods including AL + CGA [37],

Table 1: Process information.

Job	Operation	Machine									
		M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
J_1	$o_{1\text{-}1}$	2	—	4	—	10	3	—	8	—	5
	$o_{1\text{-}2}$	5	7	—	8	—	9	—	—	—	11
	$o_{1\text{-}3}$	—	6	5	—	10	9	—	—	4	—
	$o_{1\text{-}4}$	—	—	—	7	—	5	—	8	—	—
J_2	$o_{2\text{-}1}$	10	—	4	—	10	—	—	8	—	—
	$o_{2\text{-}2}$	9	—	8	—	—	7	—	6	—	—
	$o_{2\text{-}3}$	—	6	—	5	—	8	7	—	—	4
J_3	$o_{3\text{-}1}$	—	—	9	—	6	—	4	—	—	5
	$o_{3\text{-}2}$	—	—	6	—	8	—	—	7	—	—
	$o_{3\text{-}3}$	—	9	—	—	—	5	—	6	4	—
	$o_{3\text{-}4}$	—	—	4	—	3	6	—	—	—	—
J_4	$o_{4\text{-}1}$	—	8	—	5	—	—	2	4	—	—
	$o_{4\text{-}2}$	—	—	6	—	—	—	8	—	5	—
	$o_{4\text{-}3}$	5	—	—	8	—	—	4	—	—	9
	$o_{4\text{-}4}$	—	3	—	—	11	6	—	8	—	10
	$o_{4\text{-}5}$	—	—	2	—	5	8	—	—	8	—
J_5	$o_{5\text{-}1}$	6	—	—	4	—	3	—	—	8	—
	$o_{5\text{-}2}$	3	—	—	7	—	6	—	10	—	8/9
	$o_{5\text{-}3}$	—	—	9	—	—	—	4	—	—	11
	$o_{5\text{-}4}$	—	7	—	—	6	—	—	7	—	9
J_6	$o_{6\text{-}1}$	—	—	—	—	10	8	—	5	—	—
	$o_{6\text{-}2}$	—	—	—	3	—	—	2	7	—	3
	$o_{6\text{-}3}$	—	3	—	—	10	—	—	—	6	—
	$o_{6\text{-}4}$	10	—	—	6	—	10	—	—	9	8
	$o_{6\text{-}5}$	—	—	7	—	8	9	—	10	—	—
J_7	$o_{7\text{-}1}$	9	—	5	—	7	9	—	—	—	8
	$o_{7\text{-}2}$	2	—	6	—	8	—	3	7	—	—
	$o_{7\text{-}3}$	—	—	—	5	6	2	—	9	3	—
	$o_{7\text{-}4}$	9	—	15	5	3	—	—	6	—	5
J_8	$o_{8\text{-}1}$	—	3	—	6	—	8	5	—	—	—
	$o_{8\text{-}2}$	4	—	6	3	—	—	—	8	10	—
	$o_{8\text{-}3}$	—	5	8	—	—	6	—	—	5	—
	$o_{8\text{-}4}$	2	—	6	—	—	—	6	—	3	8
	$o_{8\text{-}5}$	3	—	—	5	—	3	—	5	9	8

Table 2: Comparison of the algorithm performances for the actual production instance.

Algorithm	The best solution	The worst solution	The average solution	The average computing time (s)
The basic ACO	28	36	32.4	9.327
MMAS	25	28	26.8	6.458
IACO	25	26	25.2	5.825

PSO + TS [17], PVNS [38], KBACO [19], and TSPCB [13]. Table 3 shows the best results of all these several methods, where C^* is the best value found so far.

From Table 3, it can be concluded that the proposed IACO is not worse than other algorithms, even better than several improved algorithms. Meanwhile, the proposed IACO can almost obtain the best values for the five benchmark problems for every independent test. In addition, the running time (RT) of the proposed IACO is very short; for instance, it can find the best solution for the 10×10 T-FJSP in the second iteration within 5 s.

Table 4 illustrates the comparison of the best value and the average value between IACO and those from literatures (BEDA [39]; PBABC [40]; EA [41]; and EQEA [9]) on the Kacem benchmark instances [37]. For each instance, all algorithms are run for 10 times independently. From Table 4, it can be seen easily that the best value can be obtained by these algorithms with a 100% success rate. Therefore, our IACO is very effective and robust.

4.3. Test on the BRdata Instances. Next we do tests on the ten BRdata instances [11]. Table 5 shows the comparison results

Table 3: Results of the five Kacem benchmark instances.

Problem	$n \times m$	C^*	AL + CGA [37]	PSO + TS [17]	PVNS [38]	KBACO [19]	TSPCB [13]	IACO	
								Best	RT
Case 1	4 × 5	**11**	16	—	—	**11**	**11**	**11**	0.51
Case 2	8 × 8	**14**	15	15	**14**	**14**	**14**	**14**	3.53
Case 3	10 × 7	**11**	15	—	—	**11**	**11**	**11**	3.26
Case 4	10 × 10	**7**	**7**	**7**	**7**	**7**	**7**	**7**	4.45
Case 5	15 × 10	**11**	23	12	12	**11**	**11**	**11**	4.86

Table 4: Comparison result of the best value and the average value.

Problem	$n \times m$	C^*	BEDA [39]		PBABC [40]		Chiang and Lin [41]		EQEA [9]		IACO	
			Best	AVG	Best	AVG	Best	AVG	Best	AVG	Best	AVG
Case 1	4 × 5	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**
Case 2	8 × 8	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**
Case 3	10 × 7	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**
Case 4	10 × 10	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**
Case 5	15 × 10	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**

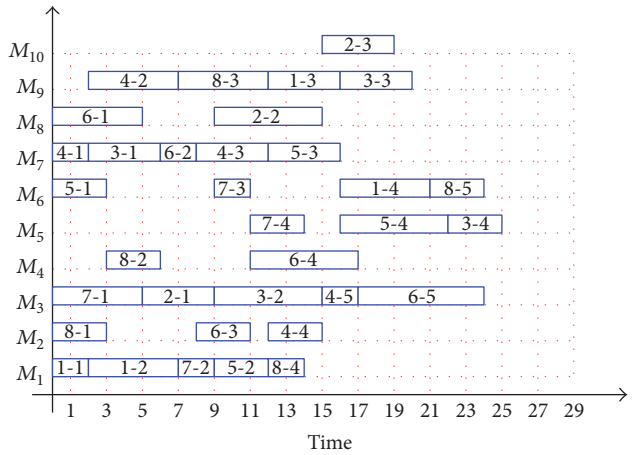

Figure 4: Gantt chart for the actual production instance obtained by IACO.

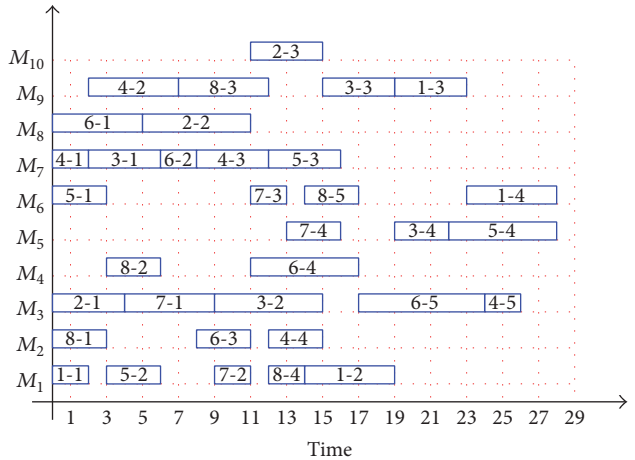

Figure 5: Gantt chart for the actual production instance obtained by the basic ACO.

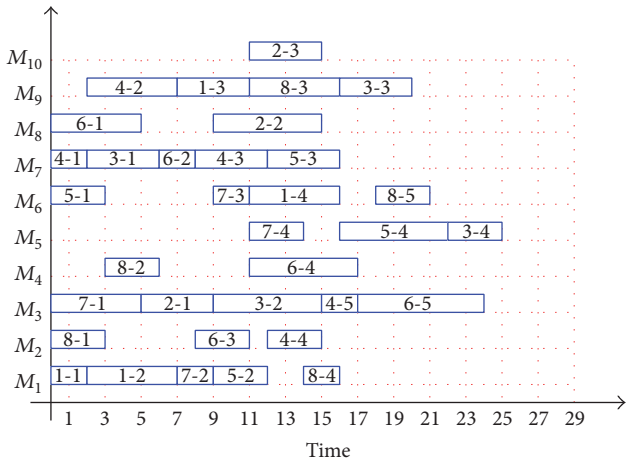

Figure 6: Gantt chart for the actual production instance obtained by MMAS.

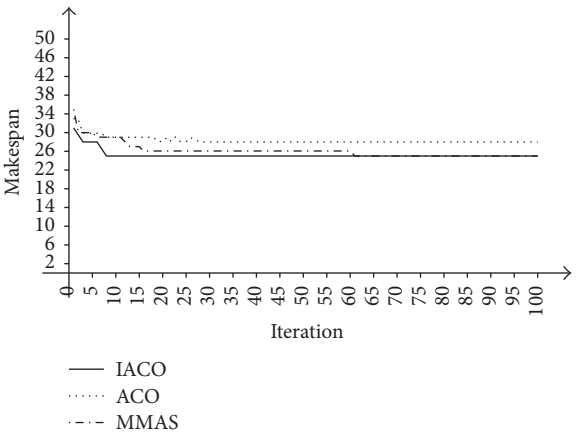

Figure 7: Convergence processes of different algorithms.

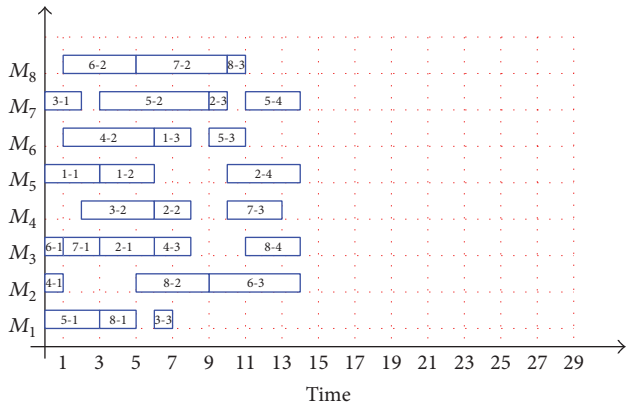

Figure 8: Gantt chart for 8 × 8 P-FJSP.

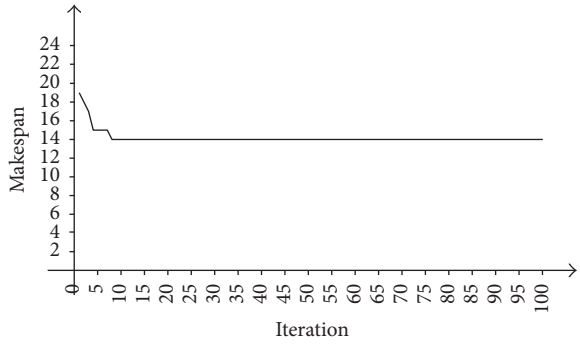

Figure 9: Convergence process for 8 × 8 P-FJSP.

Table 5: Results of the ten Kacem benchmark instances.

	LEGA [42]	GA [43]	PVNS [38]	KBACO [19]	TSPCB [13]	IACO
Mk01	40	40	40	**39**	40	40
Mk02	29	**26**	**26**	29	**26**	**26**
Mk03	—	**204**	**204**	**204**	**204**	**204**
Mk04	67	**60**	**60**	65	62	**60**
Mk05	176	173	173	173	**172**	173
Mk06	67	63	60	67	65	**60**
Mk07	147	**139**	141	144	140	140
Mk08	**523**	**523**	**523**	**523**	**523**	**523**
Mk09	320	311	307	311	310	**307**
Mk10	229	212	**208**	229	214	**208**

by using our proposed IACO and LEGA of Ho et al. [42], GA of Pezzella et al. [43], PVNS [38], KBACO [19], and TSPCB [13].

As for the best makespans, from Table 5, it can be seen that with the best results obtained our IACO is equal or smaller than that of other algorithms for dealing with almost all ten BRdata instances. Our IACO outperforms GA [40] in three out of the ten BRdata instances, outperforms KBACO [19] in six out of the ten BRdata instances, outperforms LEGA [39] in seven out of the ten BRdata instances, outperforms TSPCB [13] in four out of the ten BRdata instances, and is almost as good as PVNS [38] for the ten BRdata instances. Therefore, it is concluded that our IACO has more powerful optimizing ability in dealing with flexible job shop scheduling problem.

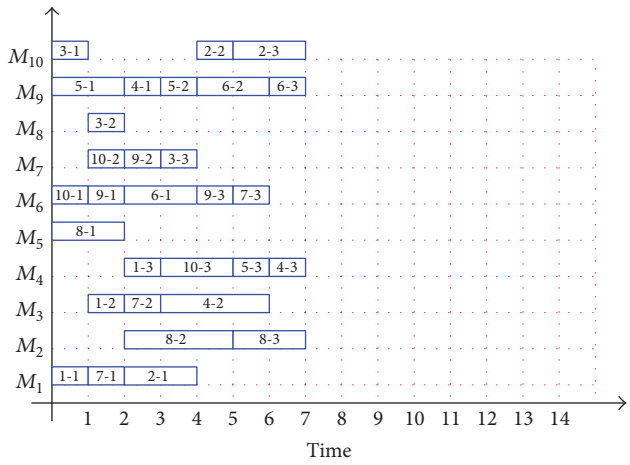

Figure 10: Gantt chart for 10 × 10 T-FJSP.

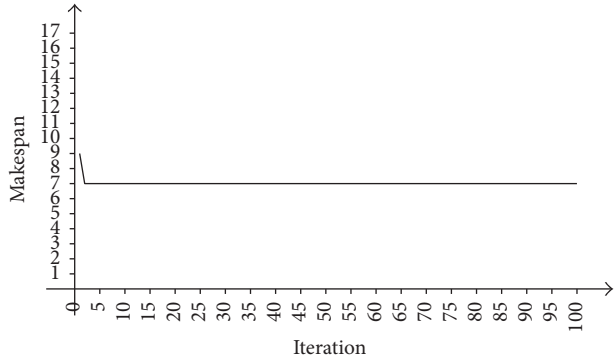

Figure 11: Convergence process for 10 × 10 T-FJSP.

5. Conclusions

In this paper, an efficient IACO is proposed for FJSP in order to minimize makespan. Experimental results on an actual production instance and two sets of well-known benchmark FJSP Instances indicate that our proposed IACO is competitive to other algorithms. The results demonstrate that the proposed IACO is effective and efficient in dealing with FJSP.

Our future work still needs to be done from the following aspects: (1) multiobjective flexible job shop scheduling problem especially related to the fuzzy due date and energy consumption need to be considered. (2) Dynamic scheduling for FJSP is another research direction, because in real manufacturing systems, unpredictable events, such as machine breakdown and new job arrivals and so on, often happen.

Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This paper is supported by the National Natural Science Foundation of China (Grant no. 51305001), Key Support Projects for Outstanding Young Talents of Anhui Province Universities (Grant no. gxyqZD2016125), Anhui Provincial Natural Science Foundation (Grant no. 1708085ME129), and Anhui Key Laboratory Open Project of Advanced Numerical Control and Servo Technology (Grant no. xjsk003).

References

[1] V. Kaplanoğlu, "An object-oriented approach for multi-objective flexible job-shop scheduling problem," *Expert Systems with Applications*, vol. 45, pp. 71–84, 2016.

[2] M. Gen and L. Lin, "Multiobjective evolutionary algorithm for manufacturing scheduling problems: state-of-the-art survey," *Journal of Intelligent Manufacturing*, vol. 25, no. 5, pp. 849–866, 2014.

[3] J. Li, Q. Pan, and S. Xie, "An effective shuffled frog-leaping algorithm for multi-objective flexible job shop scheduling problems," *Applied Mathematics and Computation*, vol. 218, no. 18, pp. 9353–9371, 2012.

[4] J.-J. Wang and Y.-J. Liu, "Single-machine bicriterion group scheduling with deteriorating setup times and job processing times," *Applied Mathematics and Computation*, vol. 242, pp. 309–314, 2014.

[5] L. Wang, S. Wang, and M. Liu, "A Pareto-based estimation of distribution algorithm for the multi-objective flexible job-shop scheduling problem," *International Journal of Production Research*, vol. 51, no. 12, pp. 3574–3592, 2013.

[6] L. Wang, G. Zhou, Y. Xu, and M. Liu, "An enhanced Pareto-based artificial bee colony algorithm for the multi-objective flexible job-shop scheduling," *The International Journal of Advanced Manufacturing Technology*, vol. 60, no. 9–12, pp. 1111–1123, 2012.

[7] L. J. Zeballos, "A constraint programming approach to tool allocation and production scheduling in flexible manufacturing systems," *Robotics and Computer-Integrated Manufacturing*, vol. 26, no. 6, pp. 725–743, 2010.

[8] I. Driss, K. N. Mouss, and A. Laggoun, "A new genetic algorithm for flexible job-shop scheduling problems," *Journal of Mechanical Science and Technology*, vol. 29, no. 3, pp. 1273–1281, 2015.

[9] X. L. Wu and S. M. Wu, "An elitist quantum-inspired evolutionary algorithm for the flexible job-shop scheduling problem," *Journal of Intelligent Manufacturing*, vol. 26, no. 2, pp. 1–7, 2015.

[10] K. Z. Gao, P. N. Suganthan, T. J. Chua, C. S. Chong, T. X. Cai, and Q. K. Pan, "A two-stage artificial bee colony algorithm scheduling flexible job-shop scheduling problem with new job insertion," *Expert Systems with Applications*, vol. 42, no. 21, pp. 7652–7663, 2015.

[11] P. Brandimarte, "Routing and scheduling in a flexible job shop by tabu search," *Annals of Operations Research*, vol. 41, no. 3, pp. 157–183, 1993.

[12] P. Brucker and R. Schlie, "Job-shop scheduling with multi-purpose machines," *Computing*, vol. 45, no. 4, pp. 369–375, 1990.

[13] J.-Q. Li, Q.-K. Pan, P. N. Suganthan, and T. J. Chua, "A hybrid tabu search algorithm with an efficient neighborhood structure for the flexible job shop scheduling problem," *International Journal of Advanced Manufacturing Technology*, vol. 52, no. 5–8, pp. 683–697, 2011.

[14] A. Jamili, M. A. Shafia, and R. Tavakkoli-Moghaddam, "A hybrid algorithm based on particle swarm optimization and simulated annealing for a periodic job shop scheduling problem," *The International Journal of Advanced Manufacturing Technology*, vol. 54, no. 1–4, pp. 309–322, 2011.

[15] J. Gao, L. Sun, and M. Gen, "A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems," *Computers and Operations Research*, vol. 35, no. 9, pp. 2892–2907, 2008.

[16] N. Al-Hinai and T. Y. Elmekkawy, "Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm," *International Journal of Production Economics*, vol. 132, no. 2, pp. 279–291, 2011.

[17] G. Zhang, X. Shao, P. Li, and L. Gao, "An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem," *Computers & Industrial Engineering*, vol. 56, no. 4, pp. 1309–1318, 2009.

[18] W. Teekeng, A. Thammano, P. Unkaw, and J. Kiatwuthiamorn, "A new algorithm for flexible job-shop scheduling problem based on particle swarm optimization," *Artificial Life and Robotics*, vol. 21, no. 1, pp. 1–6, 2016.

[19] 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," *Applied Soft Computing*, vol. 10, no. 3, pp. 888–896, 2010.

[20] J.-Q. Li, Q.-K. Pan, and M. F. Tasgetiren, "A discrete artificial bee colony algorithm for the multi-objective flexible job-shop scheduling problem with maintenance activities," *Applied Mathematical Modelling. Simulation and Computation for Engineering and Environmental Systems*, vol. 38, no. 3, pp. 1111–1132, 2014.

[21] P. Korytkowski, S. Rymaszewski, and T. Wiåniewski, "Ant colony optimization for job shop scheduling using multi-attribute dispatching rules," *The International Journal of Advanced Manufacturing Technology*, vol. 67, no. 1–4, pp. 231–241, 2013.

[22] M.-S. Lu and R. Romanowski, "Multi-contextual ant colony optimization of intermediate dynamic job shop problems," *The International Journal of Advanced Manufacturing Technology*, vol. 60, no. 5–8, pp. 667–681, 2012.

[23] C.-J. Liao, Y.-L. Tsai, and C.-W. Chao, "An ant colony optimization algorithm for setup coordination in a two-stage production system," *Applied Soft Computing*, vol. 11, no. 8, pp. 4521–4529, 2011.

[24] W. Qin, J. Zhang, and D. Song, "An improved ant colony algorithm for dynamic hybrid flow shop scheduling with uncertain processing time," *Journal of Intelligent Manufacturing*, pp. 1–14, 2015.

[25] R.-H. Huang, C.-L. Yang, and W.-C. Cheng, "Flexible job shop scheduling with due window—a two-pheromone ant colony approach," *International Journal of Production Economics*, vol. 141, no. 2, pp. 685–697, 2013.

[26] K.-L. Huang and C.-J. Liao, "Ant colony optimization combined with taboo search for the job shop scheduling problem," *Computers and Operations Research*, vol. 35, no. 4, pp. 1030–1046, 2008.

[27] B. Zhao, J. Gao, K. Chen, and K. Guo, "Two-generation Pareto ant colony algorithm for multi-objective job shop scheduling problem with alternative process plans and unrelated parallel machines," *Journal of Intelligent Manufacturing*, vol. 26, no. 3, pp. 1–16, 2015.

[28] J.-P. Arnaout, R. Musa, and G. Rabadi, "A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines—Part II: enhancements and experimentations," *Journal of Intelligent Manufacturing*, vol. 25, no. 1, pp. 43–53, 2014.

[29] C. W. Leung, T. N. Wong, K. L. Mak, and R. Y. K. Fung, "Integrated process planning and scheduling by an agent-based ant colony optimization," *Computers and Industrial Engineering*, vol. 59, no. 1, pp. 166–180, 2010.

[30] X. Li and L. Gao, "An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem," *International Journal of Production Economics*, vol. 174, pp. 93–110, 2016.

[31] L. Sun, L. Lin, Y. Wang, M. Gen, and H. Kawakami, "A Bayesian optimization-based evolutionary algorithm for flexible job shop scheduling," *Procedia Computer Science*, vol. 61, pp. 521–526, 2015.

[32] M. Mirabi, "Ant colony optimization technique for the sequence-dependent flowshop scheduling problem," *The International Journal of Advanced Manufacturing Technology*, vol. 55, no. 1–4, pp. 317–326, 2011.

[33] C. Blum, "Ant colony optimization: introduction and recent trends," *Physics of Life Reviews*, vol. 2, no. 4, pp. 353–373, 2005.

[34] Q. Ding, X. Hu, L. Sun, and Y. Wang, "An improved ant colony optimization and its application to vehicle routing problem with time windows," *Neurocomputing*, vol. 98, pp. 101–107, 2012.

[35] P. Surekha and S. Sumathi, "Solving fuzzy based job shop scheduling problems using GA and ACO," *Journal of Emerging Trends in Computing and Information Sciences*, vol. 1, no. 2, pp. 95–102, 2010.

[36] T. Stützle and H. H. Hoos, "MAX-MIN ant system," *Future Generation Computer Systems*, vol. 16, no. 8, pp. 889–914, 2000.

[37] I. Kacem, S. Hammadi, and P. Borne, "Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 32, no. 1, pp. 1–13, 2002.

[38] M. Yazdani, M. Amiri, and M. Zandieh, "Flexible job-shop scheduling with parallel variable neighborhood search algorithm," *Expert Systems with Applications*, vol. 37, no. 1, pp. 678–687, 2010.

[39] L. Wang, S. Wang, Y. Xu, G. Zhou, and M. Liu, "A bi-population based estimation of distribution algorithm for the flexible job-shop scheduling problem," *Computers & Industrial Engineering*, vol. 62, no. 4, pp. 917–926, 2012.

[40] J.-Q. Li, Q.-K. Pan, and K.-Z. Gao, "Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems," *The International Journal of Advanced Manufacturing Technology*, vol. 55, no. 9, pp. 1159–1169, 2011.

[41] T.-C. Chiang and H.-J. Lin, "A simple and effective evolutionary algorithm for multiobjective flexible job shop scheduling," *International Journal of Production Economics*, vol. 141, no. 1, pp. 87–98, 2013.

[42] N. B. Ho, J. C. Tay, and E. M.-K. Lai, "An effective architecture for learning and evolving flexible job-shop schedules," *European Journal of Operational Research*, vol. 179, no. 2, pp. 316–333, 2007.

[43] F. Pezzella, G. Morganti, and G. Ciaschetti, "A genetic algorithm for the flexible job-shop scheduling problem," *Computers and Operations Research*, vol. 35, no. 10, pp. 3202–3212, 2008.