Huang *et al. Journal of Cloud Computing: Advances, Systems and Applications* (2022) 11:9
https://doi.org/10.1186/s13677-022-00282-w

Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH Open Access

Improved firefly algorithm with courtship learning for unrelated parallel machine scheduling problem with sequence-dependent setup times

Xingwang Huang[1,2], Lingqing Chen[1], Yuxin Zhang[1], Shubin Su[1], Yangbin Lin[1] and Xuhui Cao[1*]

Abstract

The Unrelated Parallel Machines Scheduling Problem (UPMSP) with sequence-dependent setup times has been widely applied to cloud computing, edge computing and so on. When the setup times are ignored, UPMSP will be a NP problem. Moreover, when considering the sequence related setup times, UPMSP is difficult to solve, and this situation will be more serious in the case of high-dimensional. This work firstly select the maximum completion time as the optimization objective, which establishes a mathematical model of UPMSP with sequence-dependent setup times. In addition, an improved firefly algorithm with courtship learning is proposed. Finally, in order to provide an approximate solution in an acceptable time, the proposed algorithm is applied to solve the UPMSP with sequence-dependent setup times. The experimental results show that the proposed algorithm has competitive performance when dealing with UPMSP with sequence-dependent setup times.

Keywords: Firefly algorithm, Courtship learning, Unrelated parallel machines scheduling, Makespan, Sequence-dependent setup times

Introduction

The classical parallel machine scheduling problem (PMSP) widely exists in scientific and industrial production [1], including printed circuit board manufacturing [2], semiconductor wafer manufacturing dicing operations [3], computer multiprocessors task scheduling [4] and heterogeneous clusters scheduling [5], which has received extensive attention from researchers. However, in some applications, such as cloud computing [6] and edge computing [7], setup times such as waiting time for job sorting and job data transmission time are usually involved. In addition, due to the heterogeneity of devices, quite a few types of jobs also have sequence dependencies (such as Spark). This kind of problem is called the unrelated parallel machines scheduling problem (UPMSP) with sequence-dependent setup times [1].

In recent years, the research on UPMSP and its variants has received extensive attention. In UPMSP, the objective of optimization is usually to minimize the maximum completion time. Most of the UPMSP problems deal with the parallel scheduling, which requires the use of two or more machines. The UPMSP whose setup times depend on the sequence has the nature of non-deterministic polynomial, which makes it difficult to solve. Some precise algorithms have been proposed to solve the UPMSPs when the setup times are ignored, such as the branch and bound algorithm [8, 9], the cutting plane algorithm [9], etc. Nevertheless, due to the time-consuming characteristics, precise algorithms cannot be well applied to high-dimensional situations. For the case of high-dimensional, meta-heuristic algorithms are more considered to be adopted as they can find feasible solutions more efficiently and stably within

*Correspondence: cxh@jmu.edu.cn
[1]College of Computer Engineering, Jimei University, 185 Yinjiang Rd., Jimei District, Xiamen 361021, China
Full list of author information is available at the end of the article

an acceptable time range. However, the performance of these algorithms will decline in high-dimensional cases.

In addition, Firefly Algorithm with Courtship Learning (FACL) is a variant of Firefly Algorithm (FA) [10], which shows efficient performance in continuous numerical optimization problems, including high-dimensional cases [11]. This algorithm has the advantages of fewer hyper parameters and easy implementation. Therefore, we try to apply the FACL to solve the UPMSP with sequence-dependent setup times. However, the male and female attraction factors used by FACL decrease rapidly during search process, which makes it easy to terminate the search early and fall into the local optimal problem.

Aiming at the above problems, this work proposes an Improved Firefly Algorithm with Courtship Learning (IFACL) to solve the UPMSP with sequence dependent setup times. Firstly, we propose an effective technique for solving UPMSP with sequence-dependent setup times, that is, the proposed IFACL. Secondly, a new attraction factor based on Cauchy distribution is used to improve FACL's search ability. Moreover, we use a two-stage solution representation method in combination with the UPMSP with sequence-dependent setup time. Without introducing additional parameters, the search performance of male firefly in courtship learning stage is enhanced, so that the algorithm can make full use of the social information of population. Therefore, the ability of the algorithm to jump out of the local optimum is enhanced, and the global search performance is improved. Finally, a lot of simulation experiments have been carried out to evaluate the effectiveness of the IFACL in solving UPMSP with sequence-dependent setup times.

The remaining sections of this work are constructed as follows. "Related work" section presents the related work. The UPMSP mathematical model is given in "Mathematical modeling" section. "UPMSP based on firefly algorithm with courtship learning" section presents the proposed IFACL for UPMSP in detail. "Simulation experiment" section shows the experimental results and discussion. Finally, a conclusion is provided in "Conclusion" section.

Related work

Recent research on UPMSPs

In the past few years, some progress has been made in solving this challenging problem. Few exact search algorithms have been applied to UPMSP with sequence-dependent setup times. The model proposed by Guinet [12] is the basis of other Mixed Integer Linear Programming (MILP) methods, although optimality can only be guaranteed in small instances. In Vallada and Ruiz [13], an improved method was proposed for the weighted earliness-tardiness minimization model proposed by Balakrishnan et al. [14]. As far as we know, this model greatly reduces the number of binary variables, and has not been used in the makespan target before. Rocha et al. [15] developed a branch and bound algorithm to solve the UPMSP with the sequence-dependent setup times, which achieved better results in the cases of low dimension. It is only in recent years that optimal solutions for larger UPMS instances have been obtained. Avalos Rosales et al. [16] proposed a MILP that can effectively solve some instances of up to 60 jobs and 8 machines. In some iterative algorithms proposed by Tran and Beck [17] and Tran et al. [18], a similar MILP was once the main problem.

Since precise algorithms have not been proven effective in real-life examples of solving this problem until recently, some meta-heuristic algorithms have been developed to solve the UPMSP with sequence-dependent setup times. One of the meta-heuristic algorithms used to deal with the UPMSP with sequence-dependent setup times is the Variable Neighborhood Search (VNS) [19]. De Paula et al. [20] firstly used VNS to solve high-dimensional UPMSP with sequence-dependent setup times, which is to minimize task completion time and weighted delay. Vallada and Ruiz [21] proposed a genetic algorithm (GA) based on the local search enhanced crossover operator to solve the UPMSP, which is to minimize the task completion time. And they also considered the influence of the sequence dependent setup time between machines and jobs. Behnamian et al. [22] proposed a hybrid meta-heuristic algorithm that combines Ant Colony Optimization (ACO), Simulated Annealing (SA), and VNS algorithms to solve the UPMSP that also considers the sequence-dependent setup times, so as to minimize the task completion time. ACO is another meta heuristic algorithm that successfully applied to UPMSP with sequence-dependent set times. Arnaout et al. [23] proposed an enhanced ACO algorithm to solve the UPMSP with sequence-dependent setup times in the case of 10 machines and 120 jobs. Ezugwu and Akutsah [24] implemented an improved Salp Swarm Algorithm (SSA), and obtained the new complexity result of the sequence-dependent UPMSP algorithm. Niu Qun et al. [25] realized the parallel machine scheduling with adjustment time based on the improved clone selection algorithm, and it has a better performance compared with GA and the basic clone selection algorithm (CSA).

Standard firefly algorithm

Firefly Algorithm is a simple and efficient swarm intelligence algorithm for solving complex optimization problems in continuous search space. The flashing and attracting behavior of fireflies is crucial to their evolution. A firefly with a high brightness will attract a firefly with a darker brightness, and the brightness of each firefly depends on its light intensity, that is, the fitness of the objective function. The flashing brightness (the degree of mutual attraction) of fireflies can be expressed as follows:

$$\beta_{i,j}(r_{i,j}) = \beta_0 \times e^{-\gamma r_{i,j}^2}. \quad (1)$$

where $r_{i,j}$ is the Euclidian distance between two fireflies x_i and x_j, and β_0 represents the initial attraction factor, which is usually 1. Parameter γ is the fixed value of the light absorption coefficient, which is usually also 1. For two fireflies x_i and x_j randomly selected in the same search space, the distance $r_{i,j}$ between them is calculated as follows:

$$r_{i,j} = \|x_i - x_j\| = \sqrt{\sum_{d=1}^{D}(x_{i,d} - x_{j,d})^2}. \quad (2)$$

where D represents the dimension of the problem, and d represents the d-th dimension of the position vector of the firefly.

During the search, firefly x_j is attracted to the brighter firefly x_i and moves towards x_i. Its position is updated as follows:

$$x_j(t+1) = x_j(t) + \beta_{i,j} \times (x_i(t) - x_j(t)) + \alpha \times \epsilon. \quad (3)$$

where, t represents the number of iterations of the algorithm, α is the step size parameter, which is usually a random number with the value of $[0, 1]$, and ϵ is a random number between $[-0.5, 0.5]$.

In the optimization process of FA, all fireflies will move to the brighter firefly, and finally the firefly individuals will gather around the firefly with the highest brightness to complete the optimization process.

Mathematical modeling

Problem description

The UPMSP with sequence-dependent setup times, which is considered and proposed based on the improved firefly algorithm of courtship learning, is described as follows:

- The scheduling problem assumes that there are N available jobs to be allocated to M unrelated machines for processing at time zero, and the machines are independent of each other.
- There is no preemptive execution.
- The data set, including the time required for machine k to process assigned job j represented by $P_{j,k}$ and the sequence-dependent setup times of processing job j after job i on machine k given by $S_{i,j,k}$, are priori and determined. Among them, $i, j = \{1, 2, , N\}$, $k = \{1, 2, , M\}$, usually $S_{i,j,k} \neq S_{j,i,k}$.

Model construction

This work uses a model based on Mixed Integer Programming (MIP) [1, 3, 21] to represent the $(P_{j,k}|S_{i,j,k}|C_{max})$ model, which is convenient to minimize the maximum completion time (C_{max}). In order to describe the mathematical model of this problem, Table 1 gives some symbol definitions.

Table 1 Symbol definition

Symbol	Definition
C_j	The completion time of the last job j on the machine
C_{max}	The maximum completion time, namely makespan
$x_{i,j,k}$	When job j is processed immediately after job i is processed on machine k, it is 1; otherwise 0
$x_{0,j,k}$	If machine k processes job j first, it is 1; otherwise 0
$x_{i,0,k}$	If job i is the last one to be processed on machine k, it is 1; otherwise 0
V	A sufficiently large positive number
$S_{i,j,k} + P_{j,k}$	Corrected machine processing times matrix

Based on the symbol definition in Table 1 above, this article establishes a mathematical model of $(P_{j,k}|S_{i,j,k}|C_{max})$, as shown below.

$$\text{minimize } C_{max} \quad (4)$$

$$\text{s.t.} \sum_{k}^{M}\sum_{\substack{i=0 \\ i\neq j}}^{N} x_{i,j,k} = 1, \ \forall j = 1, \cdots, N; \quad (5)$$

$$\sum_{j}^{N} x_{0,j,k} = 1, \ \forall k = 1, \cdots, M; \quad (6)$$

$$\sum_{\substack{i=0 \\ i\neq h}}^{N} x_{i,j,k} - \sum_{\substack{j=0 \\ j\neq h}}^{N} x_{h,j,k} = 0, \ \forall h = 1, \cdots, N; \quad (7)$$

$$C_j - \left[C_i + \sum_{k=1}^{M} x_{i,j,k}(S_{i,j,k} + P_{j,k}) + V\left(\sum_{k=1}^{M} x_{i,j,k} - 1\right)\right] \geq 0, \ \forall i = 0, \cdots, N; \quad (8)$$

$$C_j \leq C_{max}, \ \forall j = 1, \cdots, N; \quad (9)$$

$$C_0 = 0; \quad (10)$$

$$C_j \geq 0, \ \forall j = 1, \cdots, N; \quad (11)$$

$$x_{i,j,k} \in \{0, 1\}, \ \forall i = 0, \cdots, N; \ \forall j = 0, \cdots, N; \ \forall k = 1, \cdots, M. \quad (12)$$

where, the goal of formula (4) is to minimize the maximum completion time, the constraint in formula (5) ensures that the job will be executed only once, the constraint (6) indicates that the number of jobs to be processed first by each machine is 1, the constraint in formula (7) ensures that each job is processed first or constructed into a tight post-processing job of other jobs, the formula (8) is used to calculate the completion time of each job, and the formula (9) defines C_{max} as having to be greater

than the completion time of any other job. The constraint (10) ensures that the completion time of virtual job 0 is 0, the constraint (11) ensures that the completion time of the job is not negative, and the constraint (12) defines the value range of decision variable x.

UPMSP based on firefly algorithm with courtship learning

Firefly algorithm with courtship learning

Given that fireflies attract mates in nature by glowing, interaction and information sharing between males and females is important. However, the standard FA algorithm does not distinguish the gender of individuals in the population and cannot effectively use the gender information of fireflies, which restricts the global search of FA in the presence of a large number of local extreme points.

FACL is based on the original FA algorithm, by adding a courtship learning mechanism to promote population information interaction and communication, and guide fireflies to fly, thereby enhancing the algorithm's ability to jump out of the local optimum. The courtship learning process is: the male fireflies in the population select their mates from the female firefly's external archive (A) and generate better solutions to improve the performance of the algorithm. The four key aspects of this mechanism are described as follows:

(1) Scaling mechanism. In order to make full use of the female firefly information in external archive A, individuals with lower fitness are more likely to be selected from external archive A. In this section, each female firefly in external archive A is scaled according to its fitness. The proportion transformation process of each female firefly is as follows:

$$R_i = \frac{1}{f(X_i)}, \ \forall\, i = 1, 2, \cdots, Np. \tag{13}$$

where Np represents the archive size, which can be the same size as the male firefly population; $f(X_i)$ represents the fitness of the i-th female firefly. Therefore, the female firefly with the lowest fitness in the external archive A has the largest estimation standard.

(2) Selection probability. After scaling the females in external archive A, the selection probability is designed to select the females. The selection probability is calculated as follows:

$$p_i = \frac{R_i}{(\sum_{j=1}^{Np} R_j)}, \ \forall\, i = 1, 2, \cdots, Np. \tag{14}$$

where p_i represents the probability of selecting the i-th female firefly. The selection mechanism ensures that female firefly individuals with lower fitness have higher selection probability.

(3) Female individual selection. If the selection is always based on the value order without the probability selection, FA is easy to fall into the local optimum, and it is difficult to achieve the global optimum search. Therefore, after calculating the selection probability of each female firefly by formula (14), the roulette strategy is used to select the female firefly individual to avoid the local optimum.

(4) New movement formula. According to formula (1), the attraction factor $\beta_{i,j}$ decreases with the increase of distance. In FA, when the brightness of the selected male firefly x_j is higher than the brightness of the current male firefly x_i, there will be no movement operation in this iteration; When the brightness of the selected male firefly x_j is lower than the brightness of the current male firefly x_i, the current male firefly x_j will perform a movement operation according to Eq. (15). However, when the distance between the two fireflies is large, the attractive force is extremely low (as shown in Fig. 1a, which can easily lead to the premature termination of the movement process, and an ideal solution may not be obtained. For this reason, a new movement operation is adopted in FACL to handle this situation, which is defined as follows:

$$x_j(t+1) = x_j(t) + v_{k,j} \times (x_k(t) - x_j(t)) + \alpha \times \epsilon. \tag{15}$$

where x_k is the currently selected female firefly individual, and x_j is the currently selected male firefly individual with higher brightness. v is the newly defined male and female attractiveness factor, which is defined as follows:

$$v_{k,j} = \left(\frac{r_{k,j}}{800}\right) \times \text{logsig}\left((-\beta_{k,j})^{\frac{t}{600}}\right) \times e^{-\gamma r_{k,j}^2}. \tag{16}$$

In the above formula, logsig(.) represents a logistic regression function with a range of $[0, 1]$, and t represents the number of iterations. In formula (15), v is used to replace the parameter β in formula (3) to maintain the attractiveness between male and female firefly individuals within a certain distance r, which improves the probability of finding a better solution. The male and female attractiveness factor v defined by formula (16) is shown in Fig. 1b, where, based on the consideration of control variables, the maximum number of iterations of 1000 is taken as the value of t.

Improved firefly algorithm with courtship learning

A careful analysis of the variation curve of male and female individual attraction factor v introduced in FACL algorithm with the distance r between male and female fireflies in Fig. 1b shows that the substitution of v to some extent improves the situation that β decreases rapidly to nearly 0 with the increase of r and the attraction between fireflies decreases rapidly. However, when r increases to a certain extent (near r=1), at this time, the attraction factor of male and female will decrease again rapidly, and the

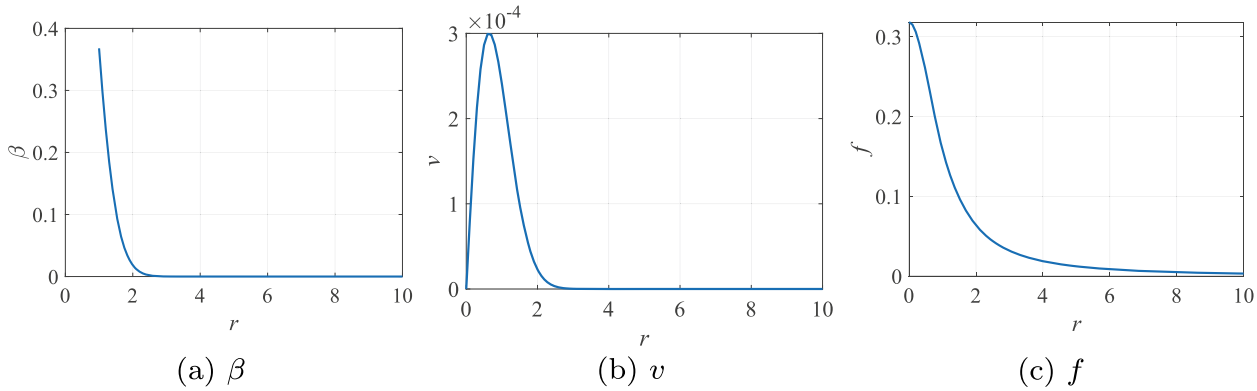


Fig. 1 The relationship between the three different attracting factors and distance r. For Fig. 1a, $\beta_0 = 1, \gamma = 1$; for Fig. 1b, $\beta_0 = 1, \gamma = 1, t_{max} = 1000$

information interaction between male and female individuals is no longer possible. What's more, compared with the random term ($\alpha \times \epsilon$) in formula (15), v is actually a very small value, which makes it difficult for male and female fireflies to interact and exchange information. Hence, the random term $\alpha \times \epsilon$ has a greater impact on the position update process, which degenerates into a random walk operation and is not conducive to the fast search of the optimal solution, especially in the discrete case. Therefore, although FACL still has a certain ability to jump out of the local optimum, it cannot search for the optima quickly.

Based on the above analysis, this work proposes a firefly algorithm based on FACL framework to improve male and female attraction factors. The algorithm introduces the Cauchy distribution function as the new attractiveness factor f to ensure that a certain attractiveness can be guaranteed when the distance r is far, so that the male firefly individual can still maintain a certain information interaction ability with the female firefly individual, continue to move in courtship behavior to ensure a certain ability to jump out of the local optimum, thereby improving the global search ability. This algorithm not only maintains the original simple structure of FACL, but also improves the accuracy of optimization.

As a new attraction factor, Cauchy distribution function has the characteristics similar to Gaussian distribution, but the peak value at the origin of Cauchy distribution is smaller and the distribution at both ends is longer. This characteristic is easy to generate random numbers with a large distance from the origin, so as to avoid the situation that the attraction rapidly decreases to nearly 0 when r is too large [26]. In this way, the individual male firefly can keep the interaction and movement of population information towards the female when the distance is large, and can jump out when trapped in the local optimum. The new attractiveness factor f is defined as follows:

$$f = \text{Cauchy}(0, 1). \tag{17}$$

where Cauchy(0,1) represents a random variable subject to the standard Cauchy distribution, defined as follows:

$$f_{k,j} = \frac{1}{\pi(r_{k,j}^2 + 1)}. \tag{18}$$

According to the above definition, the curve of the attraction factor f of male and female fireflies in the IFACL with the distance r between the male and female fireflies will be a shape with a wide distribution, a gentle falling speed, and a non-zero end. Both the FA attraction factor β and the male and female attraction factor v in FACL have a better chance of searching for a better solution. The change curve of f is shown in Fig. 1c.

Therefore, by introducing the new male and female attraction factor f into Eq. (15) to replace v, the movement formula of male firefly courtship learning from female firefly in IFACL algorithm is as follows:

$$x_j(t+1) = x_j(t) + f_{k,j} \times (x_k(t) - x_j(t)) + \alpha \times \epsilon. \tag{19}$$

Based on the above definition, the complete search process of the IFACL is illustrated as Algorithm 1.

The implementation of this additional female archiving mechanism in this article is simple. When it is time to update the female archive, choose the brighter firefly as the new female firefly to replace the original female firefly.

IFACL is applied to UPMSP with sequence-dependent setup times

Based on the literature [1], this work designs a two-stage individual representation scheme to deal with this problem. This work applies the proposed IFACL algorithm to solve the UPMSP with sequence-dependent setup times.

a)Machine assignment

The first stage is the machine assignment process. This process describes the feasible solution of assigning N tasks to M unrelated parallel machines so as to minimize the maximum completion time (C_{max}) of all machines, which can be represented by an integer vector Seq_1 whose

Algorithm 1 Improved firefly algorithm with courtship learning.

```
Randomly generate Np male fireflies as the initial population {X_i|i = 1, 2, ··· , Np};
Calculate the brightness value fb(.) of each male firefly;
Initialize the external archive A = {X_i|i = 1, 2, ··· , Np}, selection probability p, t=0;
while (t < t_max) do
  for i = 1 : Np do
    for j = 1 : Np do
      if fb(X_j) < fb(X_i) then
        Update the position of the male firefly according to Eq. (3);
        Update the optimal solution;
      else
        Calculate the selection probability using Eq. (14);
        Use the roulette strategy to calculate the brightness of the selected female firefly X_k in the external archive;
        if fb(X_k) < fb(X_j) then
          Move male firefly X_j towards female firefly X_k according to Eq. (19);
          Update the optimal solution;
        end if
      end if
    end for
  end for
  Update the female archive A;
  Update the female firefly selection probability p according to Eq. (14);
  t = t + 1;
end while
```

dimension is equal to the number of jobs. Taking the assignment of 10 jobs and 3 parallel machines as an example, if the vector Seq_1 =[3, 1, 3, 2, 1, 2, 2, 3, 1, 1], it means that the jobs {2, 5, 9, 10} will be performed on machine 1, the jobs {4, 6, 7} will be performed on machine 2, and the jobs {1, 3, 8} will be performed on machine 3. This requires the IFACL algorithm to adopt floor(.) in the iterative optimization process, where floor(.) is a function used to convert a real value to an integer value (i.e., floor(2.4) is 2). Hence, the position information of fireflies in FACL is discretized by the floor(.) function.

b)Sequence scheduling

The second stage is to determine the sequence of jobs on each machine. The sequence can be represented as a matrix of the same length as the machine-assigned vector. Therefore, the job sequence represented by Seq_2 can be expressed as a $M \times N$ matrix, which illustrates the sequence of operations on each machine. For the same previous example, the following instance of Seq_2 matrix, where extending on the Seq_1 vector, denotes that the sequence of operations on machine 1 is: job 9, job 5, job 10 and job 2; the sequence of operations on machine 2 is: job 4, job 7 and job 6; and the sequence of operations on machine 3 is: job 3, job 1 and job 8. The zero after job 2 in the first row denotes that job 2 is to be the last one to be performed by machine 1. Using the same convention, jobs 6 and 8 are the last jobs performed by machines 2 and 3, respectively.

$$Seq_2 = \begin{bmatrix} 9 & 5 & 10 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 7 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 1 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{20}$$

From the definition of Seq_2 and $x_{i,j,k}$, the value of $x_{i,j,k}$ can be easily obtained. Then, we can calculate the completion time of the jobs at the machines C_j according to formula (8). This is performed by using a large positive number ($V = \infty$), where $\sum_{k=1}^{M} x_{i,j,k} = 1$ when the jth job is processed after the ith job, so $V\left(\sum_{k=1}^{M} x_{i,j,k} - 1\right) = 0$ and $C_j = C_i + S_{i,j,k} + P_{j,k}$.

In this work, for simplicity, the corrected processing times $AP_{i,j,k}$ is used to replace the times of the two independent stages (machine processing times $P_{j,k}$ and sequence-dependent setup times $S_{i,j,k}$) to determine the maximum completion time. For the same example above, in Tables 2, 3, 4, 5, the input parameters related to the example are illustrated as an instance of the problem under study.

$AP_{i,j,k}$ can be expressed mathematically as follows:

$$\forall\, j = 1, 2, \cdots, N, \forall\, k = 1, 2, \cdots, M: \quad AP_{i,j,k} = S_{i,j,k} + P_{j,k}, \quad \forall\, i = 1, 2, \cdots, N. \tag{21}$$

Therefore, by introducing the concept of $AP_{i,j,k}$ into Eq. (4), the optimization objective function of UPMSP with sequence-dependent setup times can be given by:

$$fb(.) = C_{max} = \max_{1 \le j \le N} \{C_j\} = \max_{\substack{k=1,\cdots,M; \\ i=1,\cdots,N}} \sum_{j=1}^{N} AP_{i,j,k}. \tag{22}$$

Table 2 Example of processing times for an instance of 10 jobs and 3 machines

$P_{j,k}$	job_1	job_2	job_3	job_4	job_5	job_6	job_7	job_8	job_9	job_{10}
$Machine_1$	55	64	77	98	98	58	99	98	74	90
$Machine_2$	57	71	96	90	98	83	52	92	97	84
$Machine_3$	88	87	70	83	59	85	52	64	52	55

Table 3 Example of sequence-dependent setup times on machine 1

$S_{i,j,1}$	job_1	job_2	job_3	job_4	job_5	job_6	job_7	job_8	job_9	job_{10}
job_1	91	74	75	98	67	64	56	80	91	93
job_2	85	72	98	77	60	88	78	63	77	54
job_3	66	82	67	57	63	88	73	83	100	70
job_4	98	85	79	57	81	69	51	84	54	63
job_5	52	88	61	63	74	78	67	87	72	90
job_6	72	64	88	92	68	54	58	73	55	72
job_7	69	84	63	63	92	53	90	54	98	96
job_8	88	83	75	91	79	77	66	61	50	59
job_9	90	58	85	62	77	89	76	96	89	63
job_{10}	59	56	95	96	96	97	58	58	91	57

Table 4 Example of sequence-dependent setup times on machine 2

$S_{i,j,2}$	job_1	job_2	job_3	job_4	job_5	job_6	job_7	job_8	job_9	job_{10}
job_1	57	54	74	57	58	59	65	97	60	99
job_2	93	62	67	97	82	68	75	94	61	72
job_3	79	56	95	98	87	81	76	78	59	56
job_4	77	59	68	79	82	89	91	81	61	63
job_5	57	62	56	53	73	54	90	79	72	70
job_6	93	71	89	62	77	96	82	60	66	80
job_7	81	52	69	68	65	89	69	65	96	63
job_8	68	95	62	91	87	74	91	74	72	80
job_9	76	97	70	51	59	72	77	62	59	86
job_{10}	70	75	55	52	84	72	68	92	95	61

Table 5 Example of sequence-dependent setup times on machine 3

$S_{i,j,3}$	job_1	job_2	job_3	job_4	job_5	job_6	job_7	job_8	job_9	job_{10}
job_1	56	87	81	63	95	80	58	55	73	70
job_2	65	74	84	67	95	81	99	91	72	76
job_3	66	79	70	84	67	93	86	91	91	71
job_4	71	62	68	57	85	90	75	86	54	83
job_5	75	73	99	86	60	79	74	57	57	81
job_6	54	98	52	55	52	59	53	83	59	65
job_7	63	77	94	83	87	62	84	76	70	72
job_8	90	76	96	75	75	94	52	99	92	51
job_9	51	62	90	89	74	51	54	82	90	99
job_{10}	96	74	55	86	95	74	76	90	53	58

The target for solving UPMSP with sequence-dependent setup times problem is to minimize this objective function $fb(.)$.

The application of the algorithm

After the encoding process is completed, the UPMSP with sequence-dependent setup times is solved through the optimization steps of IFACL algorithm described

Algorithm 2 Psuedocode of the IFACL for UPMSP with sequence-dependent setup times.

1: Set the number of jobs N and the number of machines M based on the given problem.
2: Determine the number of male fireflies Np, the other parameters for IFACL.
3: Specify the initial integer value for the initial population of Np male fireflies X with dimension N. Each dimension of every male firefly is uniformly generated from $[1, M]$ and then converted to integer value with floor(.) function. Each male firefly denotes a possible scheduling solution S for the current problem.
4: Calculate the fitness value $fb(.)$ of each male firefly in the initial population.
5: Set the external archive $A = \{X_i | i = 1, 2, \cdots, Np\}$, selection probability p, t=0;
6: **while** $(t < t_{max})$ **do**
7: **for** $i = 1 : Np$ **do**
8: **for** $j = 1 : Np$ **do**
9: **if** $fb(X_j) < fb(X_i)$ **then**
10: Update the position of the male firefly according to Eq. (3);
11: Update the optimal solution;
12: **else**
13: Calculate the selection probability using Eq. (14);
14: Use the roulette strategy to calculate the brightness of the selected female firefly X_k in the external archive;
15: **if** $fb(X_k) < fb(X_j)$ **then**
16: Move male firefly X_j towards female firefly X_k according to Eq. (19);
17: Update the optimal solution;
18: **end if**
19: **end if**
20: **end for**
21: **end for**
22: Update the female archive A;
23: Update the female firefly selection probability p according to Eq. (14);
24: $t = t + 1$;
25: **end while**
26: Output the approximate optimal scheduling solution.

in "IFACL is applied to UPMSP with sequence-dependent setup times" section. In the initial stage, the IFACL algorithm first randomly assigned N planned jobs to M available processing machines to generate an initial male firefly population (represented by matrix X) consisting of Np male firefly individuals. Each male firefly individual corresponds to a job sequence on a particular machine selected after encoding in the $M \times N$ matrix described in "IFACL is applied to UPMSP with sequence-dependent setup times" section. Female firefly populations in external archive A can be initialized by male firefly populations.

The specific steps of IFACL for the UPMSP with sequence dependent setup times are as follows:

Step 1: Initialization. According to the mathematical model of UPMSP and the given range of machine processing times and sequence-dependent setup times, the parameters of IFACL are initialized.

Step 2: Population evolution. The population evolution process of IFACL is described in "Improved firefly algorithm with courtship learning" section.

Step 3: When the evolution meets the stop condition, the relevant information of the optimal female firefly is output as the approximate optimal scheduling scheme.

The psuedocode of the IFACL for UPMSP with sequence-dependent setup times is illustrated in Algorithm 2. Figure 2a and 2b are respectively the scheduling scheme result diagram and corresponding objective function variation curve obtained by IFACL after 50 iterations when processing 10 planned jobs and assigning them to 3 available processing machines according to the above steps. As can be seen from Fig. 2b, IFACL has a high convergence rate in the case of small scale.

Computational complexity

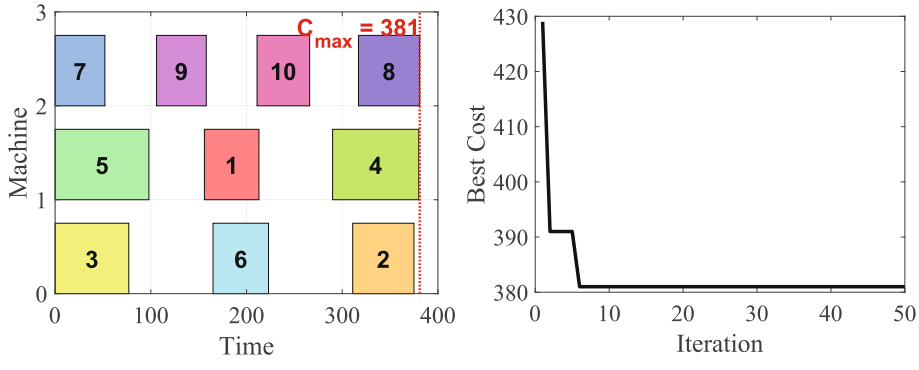
Literature [11] has demonstrated that the CL framework of FACL does not increase the method's computational complexity significantly. The IFACL algorithm is directly derived from the FACL algorithm, but the attraction factor is different, so its complexity is similar to that of FACL. Here, we analyze the time complexity of IFACL according to Algorithm 2. The related symbols are defined as follows: the population size, the number of jobs (problem dimension) and the total number of iterations are denoted as Np, N and t_{max}, respectively. First, in the initialization stage, the main computation cost of IFACL is $O(Np \times N) + O(Np)$, which is obtained by step 1-4. Then, during the optimization process in IFACL, since the average number of attractions for each firefly is $(Np - 1)/2$ [27], the computation cost of the while loop in IFACL is $O(t_{max} \times Np \times Np \times N)$. Therefore, the overall computation cost of IFACL is given as follow.

$$\begin{aligned} O(\text{IFACL}) &= O(Np \times N) + O(Np) + O(t_{max} \times Np \times Np \times N) \\ &\approx O(t_{max} \times Np \times Np \times N) \end{aligned} \tag{23}$$

Compared with other heuristic algorithms, such as PSO, whose computational complexity is $O(t_{max} \times Np \times N)$, the IFACL and FA variants have higher computational complexity.

Simulation experiment

In order to verify the effectiveness of the IFACL algorithm in solving the UPMSP with sequence-dependent setup times, two benchmark test datasets were used to verify the results. The first benchmark test dataset (named Dataset1) is generated by ourselves based on the relevant literature [28], which has been applied in many related studies [3, 21, 24, 25, 28, 29]. On this dataset, the performance of the IFACL algorithm in the $(P_{j,k}|S_{i,j,k}|C_{max})$ problem is compared with some existing meta-heuristics algorithms in the literature, including GA [21], SA [28], basic FA and FACL algorithms. In order to avoid the randomness of the test results, each algorithm was repeated



(a) The diagram of scheduling scheme (b) Objective function variation curve

Fig. 2 IFACL solves the UPMSP problem through 50 iterations

Table 6 Comparison between IFACL algorithm and four comparison algorithms in *P*, *SBalanced* instances on Dataset1

M	*N*	GA		SA		FA		FACL		IFACL (Ours)	
		Mean	*SD*	*Mean*	*SD*	*Mean*	*SD*	*Mean*	*SD*	*Mean*	*SD*
	20	1206.00	10.97	**1203.07**	**5.31**	1208.47	10.42	1205.53	11.65	1213.80	10.48
	40	**2482.13**	24.27	2534.93	**10.47**	2492.87	24.85	2496.07	23.82	2485.67	11.34
2	60	3697.73	35.04	3856.13	**15.05**	3702.93	16.24	3703.47	23.06	**3692.67**	22.53
	80	4993.20	36.92	5243.60	**24.00**	4966.93	25.66	4976.80	25.34	**4951.53**	24.43
	100	6142.40	32.25	6522.00	25.45	6128.27	28.55	6108.20	42.68	**6105.67**	**25.06**
	120	7479.33	34.91	7959.93	23.85	7399.93	**20.78**	7420.47	22.38	**7382.20**	32.67
	20	566.20	7.71	569.47	**4.05**	**564.73**	9.41	570.87	11.86	565.60	6.68
	40	1187.73	18.93	1232.47	**7.45**	1177.60	18.31	1185.27	25.84	**1167.53**	17.74
4	60	1816.73	20.95	1918.93	**13.26**	1790.47	21.69	1801.27	26.83	**1776.67**	22.85
	80	2411.73	32.17	2600.53	**14.82**	2376.00	18.81	**2371.33**	27.99	2389.20	30.86
	100	3071.13	55.10	3308.47	**13.89**	3009.13	20.76	3019.20	30.79	**2982.73**	24.60
	120	3706.87	38.60	4007.67	**15.61**	3645.13	29.08	3646.27	33.95	**3614.60**	30.84
	20	393.73	6.98	**390.27**	**4.99**	395.33	8.29	396.33	10.57	395.67	10.10
	40	805.67	16.38	834.33	**8.21**	788.47	12.98	790.27	22.43	**780.27**	14.25
6	60	1221.07	20.23	1293.73	**9.56**	**1185.20**	22.47	1198.00	23.17	1186.80	35.76
	80	1620.40	19.85	1742.07	**9.99**	1594.20	45.89	1588.20	22.43	**1565.53**	25.70
	100	2054.27	26.61	2231.00	**10.62**	2016.93	23.28	2023.87	31.50	**2008.60**	28.79
	120	1891.73	28.02	2042.67	**10.15**	1827.07	18.97	1831.33	21.93	**1798.33**	14.08
	20	295.47	9.68	294.80	**5.27**	295.40	8.51	295.27	8.44	**293.67**	8.38
	40	580.53	15.19	613.93	**6.43**	585.00	30.42	571.93	21.54	**564.00**	15.19
8	60	922.00	32.08	984.60	**5.78**	910.60	34.17	902.53	21.30	**892.60**	22.84
	80	1246.73	22.19	1331.87	**11.88**	1219.27	23.99	1220.80	34.32	**1194.13**	28.99
	100	1568.53	33.18	1693.20	**12.07**	1514.07	21.85	1519.27	23.55	**1487.13**	25.61
	120	1530.33	25.79	1665.73	**9.96**	1474.47	18.01	1480.33	24.33	**1458.27**	22.29
	20	196.40	7.84	203.27	**3.13**	209.53	36.25	199.93	31.90	212.33	35.56
	40	456.20	18.67	484.33	**7.98**	459.47	25.44	**448.00**	24.51	460.13	34.32
10	60	727.87	23.65	776.07	**7.27**	711.33	22.65	719.73	38.29	**705.00**	22.87
	80	994.93	19.96	1072.53	**7.60**	**960.47**	20.76	976.53	28.52	962.33	20.96
	100	1246.27	28.77	1363.60	**12.27**	**1205.93**	14.02	1222.07	21.59	1214.80	43.84
	120	2488.40	29.09	2701.00	**15.72**	2459.00	46.26	2433.00	33.81	**2403.93**	24.12
	20	186.07	7.10	191.40	**4.31**	188.13	9.32	**182.40**	4.93	188.20	7.73
	40	418.87	13.43	441.53	**4.73**	418.20	15.38	418.00	19.51	**412.53**	12.57
12	60	604.67	26.58	658.60	**8.24**	606.40	18.28	**602.00**	24.15	602.40	20.42
	80	831.27	15.12	907.67	**8.76**	807.40	27.49	806.87	20.68	**806.20**	20.68
	100	1073.53	39.00	1157.67	**9.69**	1040.53	49.34	1028.40	27.93	**1009.93**	27.42
	120	1294.27	43.56	1409.60	**13.28**	1241.00	24.51	1264.40	25.70	**1236.07**	42.92

for 15 times. In consideration of the need to obtain a feasible solution within an acceptable time range, the maximum number of iterations was selected as $MaxIt = 1000$, $Np = 50$, and all other parameters were directly derived from literature [24]. The second benchmark test dataset (named Dataset2) is provided by Rabadi [23], which also has been applied in many existed studies [30–33]. Dataset2 is also available via the scheduling research website (http://schedulingresearch.com/). The performance of the proposed algorithm is further verified by running

Table 7 Comparison between IFACL algorithm and four comparison algorithms in *PDominant* instances on Dataset1

M	*N*	GA		SA		FA		FACL		IFACL (Ours)	
		Mean	*SD*	*Mean*	*SD*	*Mean*	*SD*	*Mean*	*SD*	*Mean*	*SD*
	20	1964.47	14.24	**1959.67**	**4.95**	1963.40	9.19	1971.80	14.37	1971.67	11.01
	40	**3938.33**	14.92	4001.67	**12.45**	3952.80	26.55	3953.20	19.67	3943.07	21.14
2	60	**5906.93**	28.28	6078.60	17.44	5931.67	**13.65**	5923.20	32.20	5913.40	22.25
	80	7861.00	43.22	8143.07	**17.55**	7856.13	38.36	7867.47	31.88	**7847.27**	26.90
	100	10019.87	32.39	10393.20	**18.12**	9991.80	32.24	**9961.67**	33.50	9964.20	37.65
	120	11974.60	40.25	12477.33	**13.30**	11932.27	31.87	11939.47	25.83	**11929.20**	51.46
	20	955.93	7.01	953.20	**5.72**	957.60	9.74	949.00	11.75	**948.73**	12.40
	40	1920.80	15.67	1969.07	**10.55**	1915.53	12.28	1917.93	22.07	**1909.80**	14.66
4	60	2910.33	25.65	3032.53	**11.80**	**2888.67**	16.99	2902.40	31.18	2894.60	16.94
	80	3938.73	50.29	4101.80	**16.37**	3891.27	31.93	3901.60	37.11	**3880.40**	27.47
	100	4985.07	37.69	5198.87	14.47	4919.80	48.31	**4887.13**	25.86	4890.67	46.34
	120	5981.47	38.06	6302.67	**20.49**	5921.73	30.74	5906.60	35.25	**5879.87**	27.63
	20	705.73	13.25	**694.93**	**5.11**	707.60	10.48	709.53	14.56	703.53	11.53
	40	1303.20	12.55	1345.60	**6.09**	1305.27	17.60	1300.53	13.55	**1299.87**	18.62
6	60	1953.93	26.84	2036.00	**8.03**	**1920.87**	17.49	1929.53	23.60	1921.47	34.63
	80	2654.67	23.72	2773.00	19.87	**2607.67**	**13.52**	2623.20	29.42	2615.73	22.12
	100	3310.93	30.53	3479.80	**13.71**	3264.27	31.89	3255.60	26.83	**3248.93**	25.90
	120	4019.80	35.86	4210.60	**11.54**	3935.53	21.64	3938.47	34.38	**3903.67**	45.05
	20	525.00	10.09	522.60	**4.73**	**516.47**	5.18	526.67	9.80	523.33	8.07
	40	955.67	9.46	990.13	**4.96**	**937.87**	7.79	945.20	9.68	942.27	11.28
8	60	1500.80	17.82	1581.20	**8.08**	1497.67	24.37	1493.33	22.64	**1474.20**	14.91
	80	1974.27	25.86	2081.00	**10.54**	1951.67	38.40	1939.40	35.13	**1924.80**	32.64
	100	2508.87	25.33	2656.60	**11.87**	2476.07	13.59	2480.20	40.11	**2446.20**	23.06
	120	3007.47	28.56	3188.20	**18.39**	2960.93	24.16	2960.20	25.70	**2932.67**	26.43
	20	350.67	7.58	354.73	**2.91**	369.53	57.68	**346.60**	3.44	363.13	54.71
	40	746.60	10.55	786.67	**5.14**	**730.93**	11.44	772.13	58.61	753.47	46.30
10	60	1169.93	16.86	1234.40	**8.09**	**1147.93**	30.79	1155.47	27.06	1197.13	54.17
	80	1591.87	28.69	1686.00	**10.74**	1576.47	39.79	1570.80	33.85	**1562.53**	47.60
	100	2009.67	33.06	2131.67	**9.54**	1977.27	28.36	1986.00	37.41	**1976.73**	18.05
	120	2444.67	21.62	2582.20	**17.06**	2402.13	32.24	2394.93	28.54	**2369.27**	30.18
	20	337.33	9.00	342.20	**3.17**	**335.13**	5.99	337.00	5.54	337.27	6.81
	40	711.13	7.87	739.40	**4.76**	**702.33**	10.12	707.73	20.02	703.00	9.42
12	60	968.73	17.40	1034.00	**8.67**	988.53	58.71	993.27	62.44	993.60	62.19
	80	1366.20	26.09	1440.60	**6.39**	1347.67	49.73	1342.60	41.07	**1310.80**	20.34
	100	1738.47	25.03	1827.27	**9.95**	1694.00	39.96	1695.20	23.30	**1669.33**	18.41
	120	2043.00	29.06	2179.40	**24.78**	2011.27	51.51	2025.67	29.95	**1996.93**	36.05

it on Dataset2 and comparing it with the state-of-the-art work. All experiments were implemented by MATLAB 2020a programming and run in Windows environment with 8.0G RAM and 3.6GHz CPU.

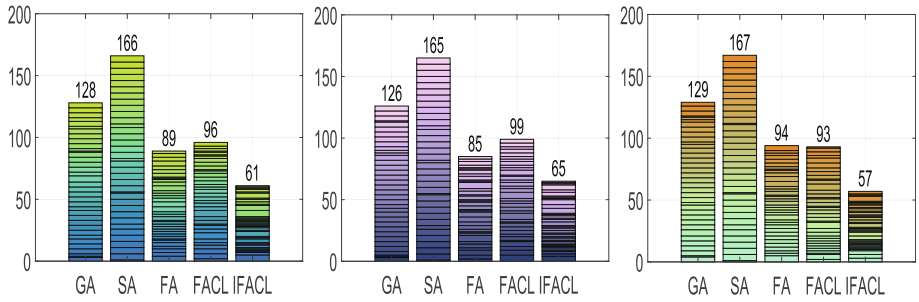
Experimental results and performance comparison on dataset1

Dataset1 contains two categories, namely processing times and setup times, and its data are randomly gen-

Table 8 Comparison between IFACL algorithm and four comparison algorithms in *SDominant* instances on Dataset1

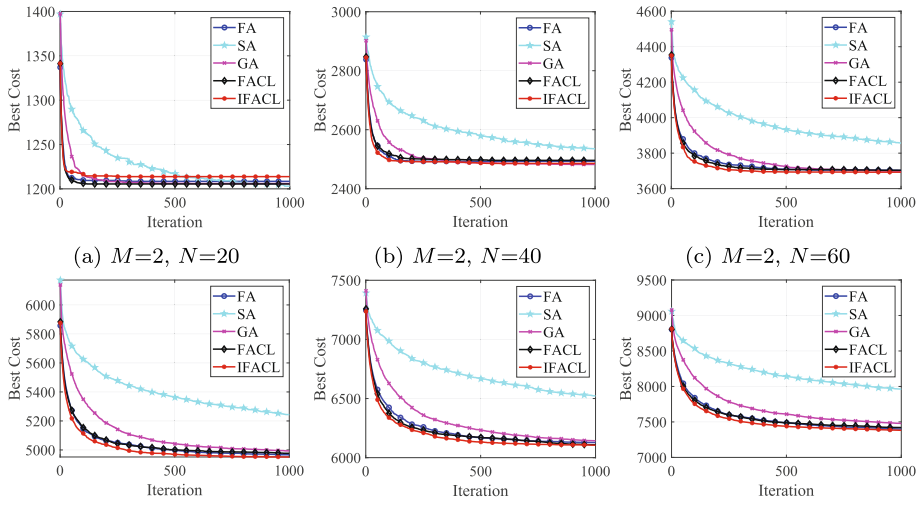
M	*N*	GA		SA		FA		FACL		IFACL (Ours)	
		Mean	*SD*	*Mean*	*SD*	*Mean*	*SD*	*Mean*	*SD*	*Mean*	*SD*
	20	1892.47	13.76	**1883.13**	**5.90**	1893.73	12.16	1886.80	12.90	1891.00	15.84
	40	**3922.13**	17.39	3985.33	**9.30**	3932.53	17.19	3936.60	24.05	3936.33	20.06
2	60	5876.07	33.47	6020.40	**16.06**	5866.93	26.32	**5859.27**	31.52	5869.80	28.91
	80	7974.93	41.16	8230.27	**22.73**	7961.53	33.63	7951.67	18.65	**7941.87**	31.74
	100	9915.27	23.09	10275.73	24.59	9875.67	**22.67**	9854.73	43.85	**9848.73**	31.46
	120	12026.87	37.70	12528.33	**23.76**	11982.53	31.11	11969.13	35.27	**11950.80**	24.97
	20	879.33	11.60	878.67	**3.96**	876.27	7.15	875.20	12.74	**873.67**	12.27
	40	1870.53	17.89	1921.93	**9.56**	1868.53	16.52	1862.27	22.58	**1852.13**	11.84
4	60	2887.60	22.82	2984.87	**13.43**	2872.60	20.21	2867.33	27.27	**2849.47**	16.52
	80	3871.73	29.04	4048.07	**16.39**	3843.67	29.12	3837.93	40.35	**3836.00**	45.45
	100	4879.60	36.38	5125.73	**19.30**	4812.07	22.24	4812.87	36.79	**4795.40**	37.98
	120	5880.93	37.62	6185.80	**16.32**	5817.47	20.91	5824.87	38.84	**5783.47**	23.91
	20	627.67	10.65	**618.73**	**3.83**	631.73	7.51	633.13	6.50	630.73	10.32
	40	1241.40	20.03	1272.60	**8.44**	1221.07	14.05	**1216.93**	16.36	1222.20	19.27
6	60	1867.47	26.16	1959.80	**10.30**	1851.33	29.41	1856.60	27.18	**1840.53**	21.91
	80	2593.80	22.92	2715.73	**16.56**	2558.33	29.94	2564.67	36.06	**2554.13**	28.27
	100	3247.13	44.29	3419.07	**16.07**	3184.67	37.71	3213.60	44.36	**3160.27**	29.00
	120	3893.80	20.51	4122.40	**15.01**	3858.07	49.06	3852.07	30.08	**3831.47**	54.53
	20	445.40	8.10	447.60	**4.45**	444.07	11.34	**443.13**	7.61	444.80	7.18
	40	881.73	20.22	914.07	**7.80**	866.67	16.22	876.00	24.57	**861.33**	17.19
8	60	1447.33	19.10	1498.00	**11.61**	**1402.40**	9.52	1413.40	21.89	1410.93	23.87
	80	1927.40	29.91	2009.33	**11.54**	1888.40	33.21	**1871.80**	25.32	1880.73	35.63
	100	2434.40	26.49	2575.00	**10.80**	2395.20	32.28	2396.87	35.87	**2375.07**	36.58
	120	2958.07	40.38	3119.80	12.37	2880.00	30.80	2891.27	30.64	**2862.60**	38.16
	20	285.40	48.23	**278.27**	**2.40**	290.47	55.17	323.87	78.35	279.87	40.64
	40	680.87	11.84	715.53	**6.37**	695.20	52.80	683.67	38.23	**673.53**	32.86
10	60	1108.13	26.79	1164.20	**5.57**	1102.27	43.02	**1085.87**	38.78	1091.60	40.61
	80	1514.67	36.83	1598.00	**13.08**	1473.93	33.18	1480.33	37.20	**1471.73**	56.10
	100	1931.60	31.36	2047.67	**11.09**	**1893.67**	34.14	1906.60	28.19	1905.00	26.88
	120	2350.53	28.88	2505.93	16.58	2312.53	28.44	2323.40	48.20	**2296.53**	35.60
	20	260.93	6.88	265.27	**3.13**	262.13	7.34	263.33	8.35	**258.87**	6.40
	40	638.13	15.89	664.53	**7.09**	638.60	18.17	639.93	22.61	**632.60**	11.44
12	60	899.93	13.74	964.87	**8.73**	922.00	58.55	919.40	54.25	924.87	61.65
	80	1291.40	28.17	1358.87	**10.98**	1244.13	13.02	1263.13	34.10	**1241.20**	16.61
	100	1640.20	13.67	1751.40	**9.46**	1616.20	45.79	1625.80	54.65	**1597.07**	26.21
	120	1954.40	31.51	2104.80	**22.65**	1963.73	62.48	**1934.07**	39.02	1938.13	28.46

erated by two discrete uniform distribution: U[50, 100] and U[125, 175]. The choice of the uniform distribution boundaries for processing times and setup times determines the level of dominance [28]. In other words, when the processing times and setup times are balanced (indicated by *P*, *S Balanced*), they are both generated by U[50, 100]. When the processing times dominate (indicated by *P Dominant*), the processing times and the setup times



(a) $P, SBalanced$ category. (b) $PDominant$ category. (c) $SDominant$ category.

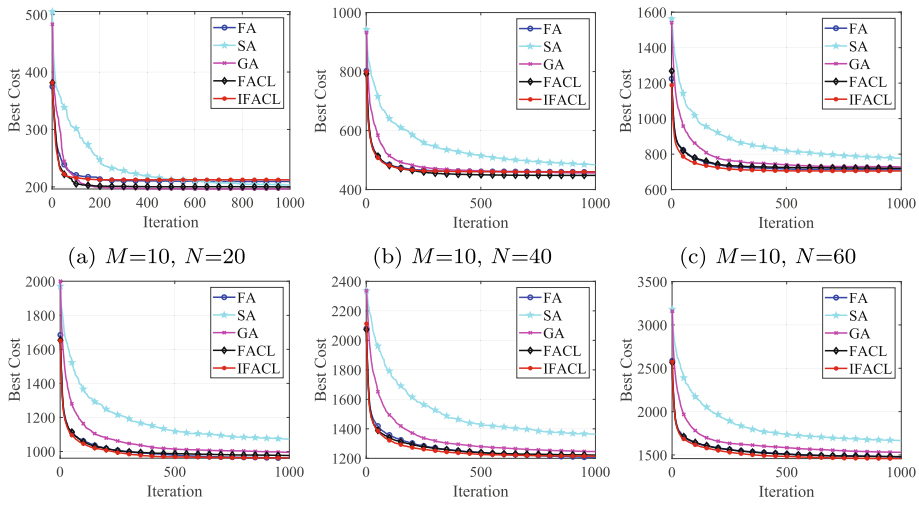
Fig. 3 Rank comparison of the five algorithms for *Mean* values over different categories on Dataset1



(a) M=2, N=20 (b) M=2, N=40 (c) M=2, N=60

(d) M=2, N=80 (e) M=2, N=100 (f) M=2, N=120

Fig. 4 Average convergence curve for each algorithm on *P, SBalanced* instances of Dataset1 when the number of machines *M*=2



(a) M=10, N=20 (b) M=10, N=40 (c) M=10, N=60

(d) M=10, N=80 (e) M=10, N=100 (f) M=10, N=120

Fig. 5 Average convergence curve for each algorithm on *P, SBalanced* instances of Dataset1 when the number of machines *M*=10

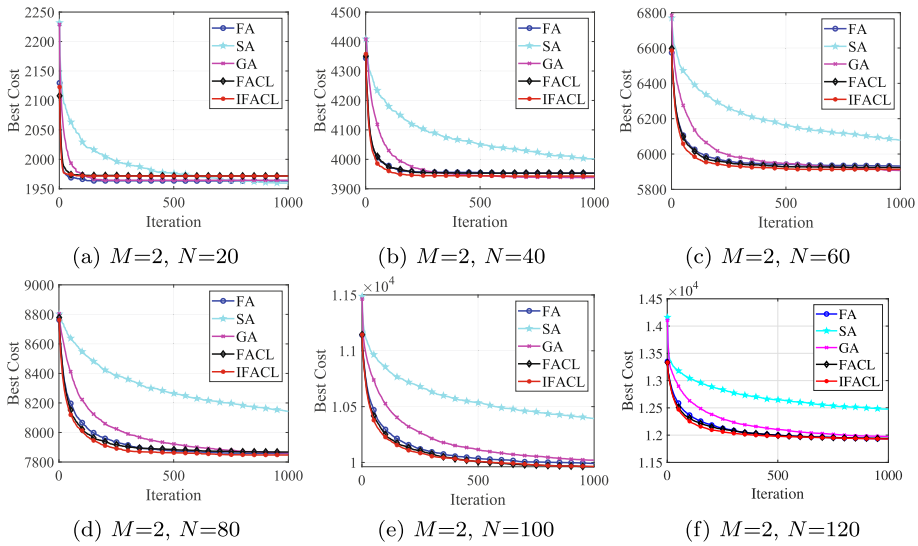


Fig. 6 Average convergence curve for each algorithm on *PDominant* instances of Dataset1 when the number of machines M=2

are generated by U[125, 175] and U[50, 100], respectively. When the setup times are dominate (indicated as *S Dominant*), the processing times and the setup times are extracted from [50, 100] and [125, 175], respectively.

The dataset generated from the above distribution consists of N jobs and M machines, covering 6 scenarios. The number of machines in each case is 2, 4, 6, 8, 10, and 12, and the number of jobs ranges from 20, 40, 60, 80, 100, and 120, respectively. Therefore, a total of 36 sets of calculation examples are constructed, and each set of calculation examples is run in each algorithm. Considering that the ratio of the computational time of algorithms to the execution time of the tasks is small, this work mainly evaluates the solving accuracy of the algorithms, which is measured by the *Mean* and variance *SD*.

Tables 6, 7, 8 illustrate the results on Dataset1 when the processing and setup times are balanced, when the processing times are dominant, and when the setup times are dominant, respectively. Among them, the optimal value of each case is marked in black and bold, and the result retains 2 decimal places.

When the processing times and setup times are balanced, according to Table 6, the results obtained by the IFACL algorithm on Dataset1 can obtain 24 optimal *Mean* values out of the 36 test cases in terms of accuracy. Therefore, in terms of accuracy, it is obvious that the IFACL is

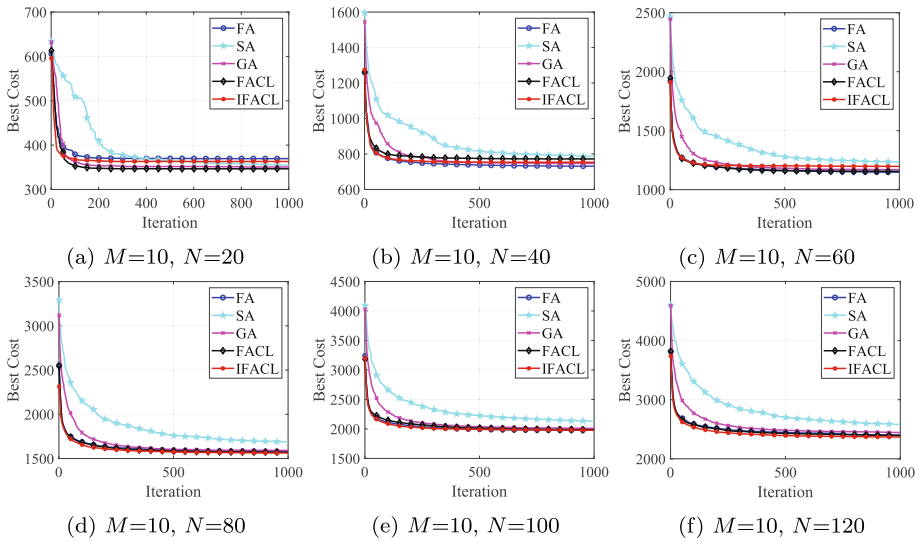


Fig. 7 Average convergence curve for each algorithm on *PDominant* instances of Dataset1 when the number of machines M=10

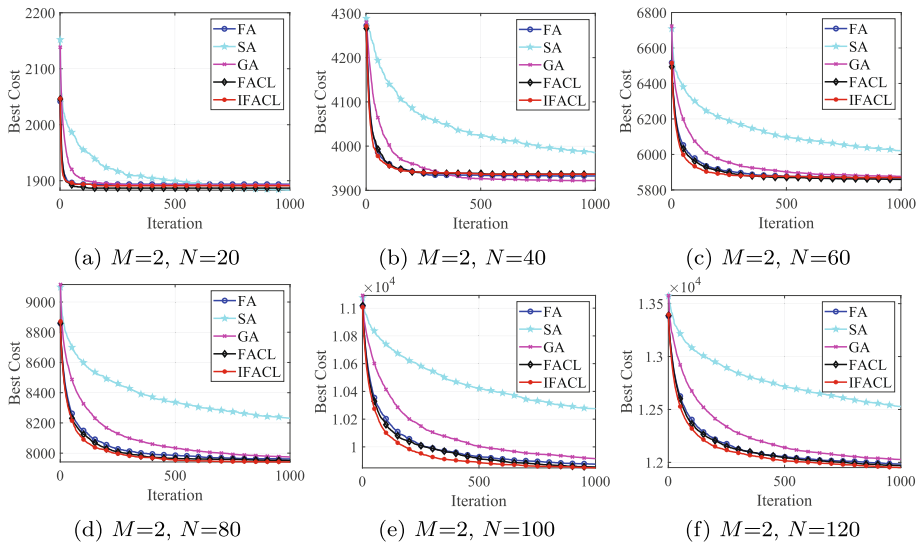


Fig. 8 Average convergence curve for each algorithm on *SDominant* instances of Dataset1 when the number of machines M=2

superior to the other four compared algorithms in terms of accuracy, followed by the other two FA variants (FA and FACL). In terms of stability, SA obtains 34 best *SD* values in the 36 cases, indicating that SA is of the highest stability. Besides that, IFACL also performs pretty good stability on Dataset1.

When the processing times dominate, as can be seen from Table 7, IFACL also shows the best performance in terms of accuracy as it achieves 19 optimal results out of the 36 test cases on Dataset1, followed by FA and FACL. Since the SA obtains 33 best *SD* values out of the 36 test cases, the SA still shows the best stability.

When the setup times dominate, according to Table 8, the IFACL achieves 23 best *Mean* values out of the 36 test cases on Dataset1, which shows that IFACL still performs best in terms of accuracy. Obviously, SA is the most stable, too.

To make the comparison of the five algorithms in terms of accuracy more intuitively and precisely, on the basis of Tables 6, 7, 8, this work also compares the rank of *Mean* values obtained by each comparison method in each of the 36 cases on three different categories of Dataset1, as is shown in Fig. 3. For each cases on different categories, the top-ranked approach will add 1 on the correspond-

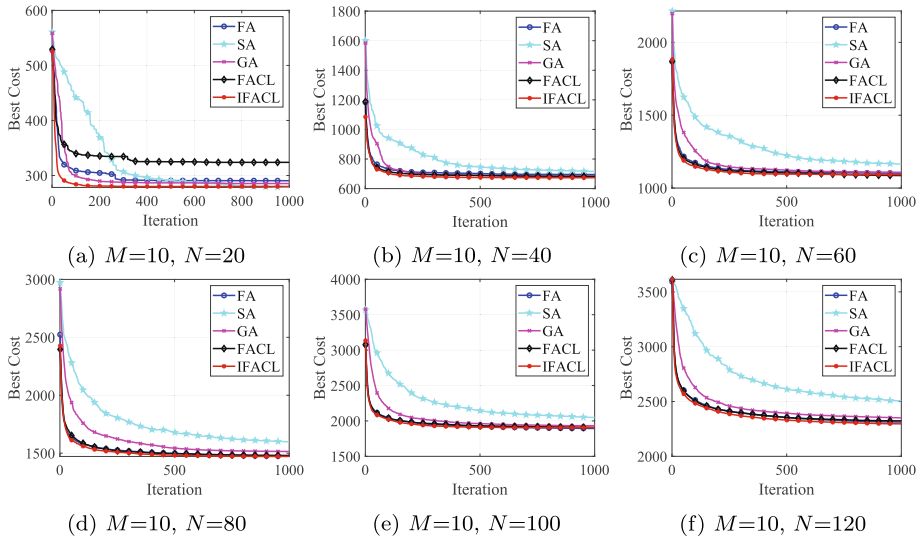


Fig. 9 Average convergence curve for each algorithm on *SDominant* instances of Dataset1 when the number of machines M=10

ing bar and the last-ranked approach will add 5, which denotes that the best method would correspond to the lowest bar. As can be seen from the Fig. 3, IFACL ranks first in three different categories, while SA ranks last. The result illustrates that when solving the UPMSP with sequence-dependent setup times on Dataset1, our proposed IFACL-based scheduling approach has best accuracy in all compared algorithms.

Figures 4, 5, 6, 7, 8 and 9 show the average convergence curves of these five algorithms on Dataset1 with different job numbers when the number of machines is 2 and 10, respectively, including the *P*, *S*, *Balanced* instances, *PDominant* instances and *SDominant* instances. These figures show that the convergence rate of IFACL is similar to that of FA and FACL, but better than that of GA and SA. In addition, these figures further demonstrate the high performance of IFACL in terms of accuracy.

Experimental results and performance comparison on dataset2

In this subsection, another UPMSP benchmark dataset named Dataset2 is used to evaluate the proposed algorithm. Dataset2 also contains the two categories, including the processing times and the setup times. In this section, the IFACL is compared with the state-of-the-art from the literature such as FSS [30], WO [31], GADP2 [32], SADP [32], IFA [33]. These approaches are chosen since they established their quality to address the UPSMP with sequence-dependent setup times and all of them adopt the same benchmark instances (that is, Dataset2) to solve UPMSPs in previous studies.

Similar to the comparison approaches, such as IFA, only show results of *P*, *SBalanced* instances on Dataset2, we also conduct experiments in *P*, *SBalanced* cases here. According to literature [30], the maximum number of iterations is set to 10000 and the size of population is set to 100. Note that the results of the comparison algorithms are derived directly from the corresponding papers.

Table 9 presents the results of the methods on Dataset2 when the processing and setup times are balanced. Since the FSS only present the *Mean* results, we also just present the *Mean* results in Table 9. Furthermore, WO, GADP2 and SADP only present the results for problem instances with 2, 6 and 12 machines and 20, 40, 60, and 80 jobs, therefore, Table 9 also provides the corresponding results for the same problem instances. Among them, the optimal value of each case is marked in black and bold.

As can be seen from Table 9, both the IFACL and the FSS obtain the 5 best *Mean* values out of 12 instances, followed by the WO and the IFA. It can also be observed that when the number of jobs is 20, IFACL manages to outperform the other approaches. When the number of jobs is 40, IFACL also performs the best or second best. This demonstrates that IFACL is suitable for processing problem instances with a relatively small number of jobs. Besides, comparing with IFA, IFACL obtains 9 better results in 12 problem instances, which also verifies the effectiveness of the introduction of Cauchy distribution as a factor of female and male attraction on the basis of the FACL framework aforementioned in this work in improving FA performance to a certain extent. Although IFACL does not perform optimally when the number of jobs is 60 or 80, it is still a valid candidate algorithm overall.

In summary, the IFACL has great potential to achieve high-quality solutions over the Dataset1 and Dataset2 of the UPMSP problems. It outperforms the FA, FACL, GA and SA in most problem instances on Dataset1, and it performs quite well on Dataset2 compared to the state-of-the-art work, especially for the problem instances with a relatively small number of jobs. Whereas, the proposed

Table 9 Comparison between IFACL and the state-of-the-art in *P*, *SBalanced* instances on Dataset2

M	*N*	*Mean*					
		FSS [30]	**WO [31]**	**GADP2 [24]**	**SADP [24]**	**IFA [33]**	**IFACL (Ours)**
	20	1234.87	1235	1254	1255	1201	**1187.40**
	40	**2391.20**	2397	2459	2462	2396	2392.60
2	60	**3563.40**	3574	3675	3764	3612	3686.40
	80	**4723.60**	4730	4872	4879	4813	4904.33
	20	445.87	446	454	455	407	**378.47**
	40	777.53	778	831	841	767	**758.67**
6	60	1128.73	1133	1246	1259	**1125**	1171.67
	80	**1501.33**	1513	1648	1662	2346	1562.33
	20	231.00	231	239	241	188	**181.67**
	40	430.93	431	455	447	493	**404.73**
12	60	562.27	**562**	649	669	715	605.93
	80	**755.40**	762	849	891	959	801.93

IFACL benefits from the advantages of both the FACL framework and the new female and male attractiveness factor, thereby improving its search performance and raising its ability to avoid getting stuck in local optima. Furthermore, compared with the FA, FACL, GA and SA, the IFACL is also the fastest algorithm to reach the minimum makespan, as shown in the convergence curves. The comparison with the state-of-the-art work demonstrates the effectiveness of the IFACL in some of the problem instances on Dataset2, and the IFACL is competitive with other algorithms. However, the proposed approach has some limitations, such as performance degradation when the number of jobs is large, as shown in the Table 9. It stands for the reason that the new attractiveness factor in IFACL is also close to 0 at the later stage of the search, so the ability of IFACL to jump out of the local optimum in high dimensions is weaker than that of some state-of-the-art works. In addition, the processing times, the initial job sequences and the sequence-dependent setup times should be generated before starting the search process.

Conclusion

Aiming at the UPMSP with sequence-dependent setup times, this work proposes the IFACL algorithm, which takes minimizing the maximum completion time as the scheduling optimization goal to solve the UPMSP with sequence dependent setup times. For the IFACL algorithm, the Cauchy distribution function with wide distribution and gentle decline is introduced as a new attraction factor of male and female. This design can enhance the social information exchange between fireflies and enhance the ability of the algorithm to jump out of the local optimum in high-dimensional situations. The test results of two benchmark test datasets show that the scheduling algorithm proposed in this work is effective in dealing with sequence-dependent setup times UPMSP. Compared with GA and SA in terms of accuracy on Dataset1, the IFACL has obvious advantages, and it is also greatly improved compared with FA and FACL. Furthermore, based on the empirical results on Dataset2, the IFACL is shown to be competitive with other state-of-the-art algorithms.

The scheduling problem of application scenarios such as edge computing is a hot topic of scheduling research. However, there is no unified model standard yet. In the future, we will continue to use the UPMSP model for reference and study under the conditions of considering more constraints such as bandwidth and different network service providers' charges, and propose efficient algorithm to solve the problem.

Acknowledgments
The authors would like to thank the anonymous referees for their valuable comments and suggestions.

Authors' contributions
XH contributed to the modeling, conducted the experiments, performed the data analysis and wrote the manuscript; LC, YZ, YL and XC contributed to analysis through constructive discussions. SS contributed to the conceptualization, writing review and visualization. The authors read and approved the final manuscript.

Funding
Project is supported by the National Natural Science Foundation of China (U1936114, 62006096), the Natural Science Foundation of Fujian Province of China (2020J01699, 2020J01700, 2020J05146, 2020J01697) and the General project of Education Department of Fujian Province (JAT190320).

Availability of data and materials
The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Ethics approval and consent to participate
The work is a novel work and has not been published elsewhere nor is it currently under review for publication elsewhere.

Consent for publication
Informed consent was obtained from all individual participants included in the study.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]College of Computer Engineering, Jimei University, 185 Yinjiang Rd., Jimei District, Xiamen 361021, China. [2]Spatial Intelligence Lab, Jimei University, 185 Yinjiang Rd., Jimei District, Xiamen 361021, China.

Received: 19 October 2021 Accepted: 25 February 2022
Published online: 12 March 2022

References

1. Ezugwu A (2019) Enhanced symbiotic organisms search algorithm for unrelated parallel machines manufacturing scheduling with setup times. Knowl-Based Syst 172:15–32
2. Chen CL, Chen CL (2008) Hybrid metaheuristics for unrelated parallel machine scheduling with sequence-dependent setup times. Int J Adv Manuf Technol 43:161–169
3. Ezugwu A, Akutsah F (2018) An improved firefly algorithm for the unrelated parallel machines scheduling problem with sequence-dependent setup times. IEEE Access 6:54,459–54,478
4. Wu L, Wang S (2018) Exact and heuristic methods to solve the parallel machine scheduling problem with multi-processor tasks. Int J Prod Econ 201:26–40
5. Orts F, Ortega G, Puertas A (2020) On solving the unrelated parallel machine scheduling problem: active microrheology as a case study. J Supercomput 76(11):8494–8509
6. Li B, Fu X, Gao X, Zhang Z (2012) Research on parallel machine scheduling problem in cloud computing based on ant colony algorithm. J Huazhong Univ Sci Technol 40:225–229
7. Huang X, Li C, Chen H, An D (2019) Task scheduling in cloud computing using particle swarm optimization with time varying inertia weight strategies. Clust Comput 23:1137–1147
8. Bülbül K, Şen H (2017) An exact extended formulation for the unrelated parallel machine total weighted completion time problem. J Sched 20(4):373–389
9. Li X, Yalaoui F, Amodeo L, Chehade H (2012) Metaheuristics and exact methods to solve a multiobjective parallel machines scheduling problem. J Intell Manuf 23(4):1179–1194
10. Yang XS, He X (2013) Firefly algorithm: recent advances and applications. Int J Swarm Intell 1(1):36–50
11. Peng H, Zhu W, Deng C, Wu Z (2021) Enhancing firefly algorithm with courtship learning. Inf Sci 543:18–42

12. Guinet A (1991) Textile production systems: a succession of non-identical parallel processor shops. J Oper Res Soc 42(8):655–671
13. Vallada E, Ruiz R (2012). Springer, Just-in-Time Systems
14. Balakrishnan N, Kanet JJ, Sridharan V (1999) Early/tardy scheduling with sequence dependent setups on uniform parallel machines. Comput Oper Res 26(2):127–141
15. Rocha PL, Ravetti MG, Mateus G, Pardalos P (2008) Exact algorithms for a scheduling problem with unrelated parallel machines and sequence and machine-dependent setup times. Comput Oper Res 35:1250–1264
16. Avalos-Rosales O, Angel-Bello F, Alvarez A (2015) Efficient metaheuristic algorithm and re-formulations for the unrelated parallel machine scheduling problem with sequence and machine-dependent setup times. Int J Adv Manuf Technol 76(9-12):1705–1718
17. Tran TT, Beck JC (2012) Logic-based Benders decomposition for alternative resource scheduling with sequence dependent setups. Front Artif Intell Appl 242:774–779
18. Tran TT, Araujo A, Beck JC (2016) Decomposition methods for the parallel machine scheduling problem with setups. INFORMS J Comput 28(1):83–95
19. Pacheco J, Porras S, Casado S, Baruque B (2018) Variable neighborhood search with memory for a single-machine scheduling problem with periodic maintenance and sequence-dependent set-up times. Knowl Based Syst 145:236–249
20. Paula MR, Ravetti MG, Mateus G, Pardalos P (2007) Solving parallel machines scheduling problems with sequence-dependent setup times using variable neighbourhood search. IMA J Manag Math 18:101–115
21. Vallada E, Ruiz R (2011) A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. Eur J Oper Res 211:612–622
22. Behnamian J, Zandieh M, Ghomi S (2009) Parallel-machine scheduling problems with sequence-dependent setup times using an aco, sa and vns hybrid algorithm. Expert Syst Appl 36:9637–9644
23. Arnaout JPM, Rabadi G, Musa R (2010) A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times. J Intell Manuf 21:693–701
24. Ewees AA, Al-qaness MA, Elaziz MA (2021) Enhanced salp swarm algorithm based on firefly algorithm for unrelated parallel machine scheduling with setup times. Appl Math Model 94:285–305
25. Qun N, Taijin Z, Xiaohai W, Hongyun Z (2012) Clonal selection algorithm for parallel machine scheduling with setup time. J SE Univ (Nat Sci Ed) 42(z1):163–167
26. Changyuan L, Yuyan R, Xiaojun B (2020) Timing optimization of regional traffic signals based on improved firefly algorithm. Control Theory Appl 35(12):2829–2834
27. Wang H, Wang W, Sun H, Rahnamayan S (2016) Firefly algorithm with random attraction. Int J Bio-Inspired Comput 8(1):33–41
28. Kim D, Kim K, Jang W, Chen FF (2002) Unrelated parallel machine scheduling with setup times using simulated annealing. Robot Comput Integr Manuf 18:223–231
29. Qiang S (2020) A hybrid multi-objective teaching-learning-based optimization algorithm for unrelated parallel machine scheduling problem. Control Theory Appl 37(10):2242–2256
30. Jovanovic R, Voß S (2021) Fixed set search application for minimizing the makespan on unrelated parallel machines with sequence-dependent setup times. Appl Soft Comput 110:107,521
31. Arnaout JP (2020) A worm optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times. Ann Oper Res 285(1):273–293
32. Chang PC, Chen SH (2011) Integrating dominance properties with genetic algorithms for parallel machine scheduling problems with setup times. Appl Soft Comput 11(1):1263–1274
33. Ezugwu AE, Akutsah F (2018) An improved firefly algorithm for the unrelated parallel machines scheduling problem with sequence-dependent setup times. IEEE Access 6:54,459–54,478

Publisher's Note