Soft Computing (2021) 25:9545–9557
https://doi.org/10.1007/s00500-021-05889-w

OPTIMIZATION

Modified whale optimization algorithm for solving unrelated parallel machine scheduling problems

Mohammed A. A. Al-qaness[1] · Ahmed A. Ewees[2,3] · Mohamed Abd Elaziz[4,5]

Accepted: 13 May 2021 / Published online: 31 May 2021

Abstract

Unrelated parallel machine scheduling problem (UPMSP) with sequence-dependent setup times is considered a hot topic among the researchers, as it presents more complexity to be able to find an optimal solution. Many efforts have been made to solve UPMSP problems and established their performances. Therefore, in this study, a new method is introduced to address UPMSP problems with sequence-dependent and machine-dependent setup time. Our proposed method utilizes two meta-heuristic techniques, the whale optimization algorithm (WOA) and the firefly algorithm (FA), by combining their features to perform this task. The hybrid model is called WOAFA. For more detail, the operators of the FA are employed to improve the exploitation ability of the WOA by serving as a local search. Moreover, the quality of the proposed WOAFA method is tested by comparing with well-known meta-heuristic algorithms over six machines and six jobs, namely (2, 4, 6, 8, 10, and 12 machines) and (20, 40, 60, 80, 100, and 120 jobs).

Keywords Whale optimization algorithm · Firefly algorithm · Meta-heuristic · Unrelated parallel machine scheduling problem · Local search

1 Introduction

Scheduling is one of the most important issues in different applications, including manufacturing and various services, such as airport schedules, train schedules, and others. Efficient scheduling is done by designating a number of jobs to a number of limited resources according to the operational restrictions. Therefore, efficient scheduling has a significant impact on such applications and fields (Lin and Ying 2014; Santos et al. 2019). Parallel machine scheduling (PMS) is widely employed in different systems, including manufacturing and other services. PMS problems have received wide attention in the past decades, and they include three well-known categories, called identical, uniform, and unrelated PMS problems (Ying et al. 2012). In the identical category, the job time is the same in all machines, whereas in uniform category, each machine has a different speed and works at constrain rates. More so, in unrelated one, each machine works at different rates, and each machine processes different assigned jobs (Afzalirad and Rezaeian 2016).

More so, the setup time of scheduling problems has two types, called, sequence-dependent and sequence-independent (Hamzadayi and Yildiz 2017). For sequence-dependent type, the setup time is depended on both jobs that already being scheduled and the last scheduled jobs, whereas in the sequence independent type, the setup time is added to the processing time of the job (Hamzadayi and Yildiz 2017; Kim and Lee 2012).

In general, the existed studies addressed the unrelated PMS problems (UPMSPs) with job sequence-dependent and

✉ Mohammed A. A. Al-qaness
alqaness@whu.edu.cn

Ahmed A. Ewees
ewees@du.edu.eg

Mohamed Abd Elaziz
abd_el_aziz_m@yahoo.com

1 State Key Laboratory for Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, Wuhan 430079, China

2 Department of e-Systems, University of Bisha, Bisha 61922, Kingdom of Saudi Arabia

3 Department of Computer, Damietta University, Damietta 34511, Egypt

4 Department of Mathematics, Faculty of Science, Zagazig University, Zagazig 44519, Egypt

5 School of Computer Science and Robotics, Tomsk Polytechnic University, Tomsk, Russia

machine-dependent setup times (JSDMDSTs) to minimize the makespan. The UPMSPs can be considered as a number of N jobs, which is required to be assigned to m machine from a set of (R_M) unrelated parallel machines, and the makespan (C_{max}) must be minimized. Each nth job has a single task which requires a processing time. More so, the JSDMDST (S_{ijk}) is addressed because it is a critical issue in this field. Therefore, there are differences between the S_{ijk} that needed for the two i and j consecutive jobs on $k, k = 1, ..., M$ machine and reverse two jobs (S_{ijk} on k machine between j and i jobs). Moreover, the S_{ijk} between i and j jobs on k machine is different from the S_{ijk} of other j jobs on other $k1$. Therefore, this is an NP hard problem and can be signified by $R_M/S_{ijk}/C_{max}$ (Lin et al. 2011).

In previous decades, many studies had been investigated to address the unrelated parallel machine scheduling problems (UPMSPs); for example, the first study had been proposed by McNaughton (1959) in 1959. Many efforts had been presented during the past decades; however, only a few solution models had been improved to solve UPMSPS without considering the JSDMDSTs, for example, a branch and bound algorithm (Rocha et al. 2008). Also, in (Helal et al. 2006), the authors introduced a mixed integer programming model (MIP) to address UPMSPs. More so, in (Rocha et al. 2008), two MIP models were proposed to solve UPMSPs and an efficient algorithm, called branch-and-bound(B&B). In (Fanjul-Peyro et al. 2019), the authors presented a MIP model and a mathematical programming to solve UPMSPs with JSDMDSTs.

Recently, meta-heuristic (MH) methods have been widely employed in various optimization problems, including UPMSPs and cloud scheduling (Attiya et al. 2020). For example, the variable neighborhood search (VNS) was employed by Pacheco et al. (2018) to deal with UPMSPs with sequence-dependent setup times, but with only one machine. The VNS also had been applied for large instances of UPMSPs with setup times in (De Paula et al. 2007). In (Logendran et al. 2007), Tabu search (TS) algorithm was applied to enhance six algorithms for solving UPMSPs. Genetic algorithm (GA) is also utilized for UPMSPs as proposed by Vallada and Ruiz (2011). Yilmaz Eroglu et al. (2014) proposed a modified GA to address the UPMSPs with setup time. In their study, the GA is enhanced by applying a local search algorithm that enhances the search power of the GA. In (Bektur and Saraç 2019), two MH methods called TS and simulated annealing (SA) with a MIP model were proposed to address UPMSPs by minimizing the tardiness. The SA also applied by (Hamzadayi and Yildiz 2016a, b) with several dispatching rules methods to address the problems of identical parallel machines. In (Pakzad-Moghaddam 2016), particle swarm optimization (PSO) was implemented to address jobs scheduling problems of the uniform parallel machines (Pakzad-Moghaddam 2016). Lin and Ying (2014) proposed an enhanced artificial bee colony (ABC) algorithm to address UPMSP problems by minimizing the makespan. Moreover, the improved ABC had been evaluated and compared with several existed methods and showed better performance. Jouhari et al. (2019) proposed a combination of the SA and sine–cosine algorithm (SCA) for solving UPMSPs with JSDMDSTs. The SCA is applied to improve the search performance of the SA, and the combined model was evaluated and compared to several MH methods, such as the originals SA and SCA, and it showed a better performance.

In (Arroyo et al. 2019), the authors presented an iterative greedy (IG) algorithm and a MIP model to solve the UPMSPs. As authors mentioned, the IG algorithm showed better performance than several MH methods, including ant colony optimization (ACO), discrete differential evolution, and SA. Mir and Rezaeian (2016) proposed a hybrid PSO and GA to solve UPMSPs by minimizing the total machine load.

Besides the performance achieved by the previous UPMSPs methods based on MH techniques, they still suffer from some limitations that affect their performance. For example, some of these methods give the exploration phase more attention than the exploitation phase. In contrast, some method made the improvement of the exploitation ability is the main target. Thus, solutions can attract to the optimal local point. In addition, according to the no free lunch theorem that assumes there is no one algorithm that can be solved all optimization problems with the same performance. So, this motivated us to present an alternative method to tackling the UPMSPs problem with JSDMDSTs. This method depends on an improved whale optimization algorithm (WOA) using firefly algorithm (FA) that is used as a local search algorithm; the proposed method is called WOAFA. The WOA is an MH algorithm presented by Mirjalili and Lewis (2016). WOA simulates behaviors of humpback whales and it has been applied in various applications, such as feature selection (Mafarja and Mirjalili 2017), image segmentation (El Aziz et al. 2017), global optimization problems (Trivedi et al. 2016), flow shop scheduling problem (Abdel-Basset et al. 2018), sentiment analysis (Akyol and Alatas 2020), gas consumption prediction (Qiao et al. 2020), and others (Mirjalili et al. 2020). The FA is also a nature-inspired MH that mimics the flashing behavior of fireflies, proposed by Yang (2009); Yang and He (2013). It has been utilized in numerous applications, such as image processing (Yang 2020), cloud computing scheduling (Rajagopalan et al. 2020), forecasting models (Zhou et al. 2019), opinion leader classification (Jain and Katarya 2019), and other applications (Nayak et al. 2020).

The proposed WOAFA is a new hybrid approach to solve UPMSPs by exploiting the power of the WOA, which is improved using the operators of the FA, where the FA is

utilized as the local search method for the WOA. Therefore, the proposed WOAFA starts by generating random individuals to represent UPMSPs solutions. The dimension of each individual is represented by job numbers, and each individual value represents the index of the machine that executes a corresponding job. Moreover, to determine the best individual solution, the fitness function is measured to evaluate each individual solution. Thereafter, the probability fitness value of each individual is computed. Therefore, the individual solutions will be updated based on the computed probability by exploiting WOA or FA operators. Thus, previous steps are executed until meeting the terminal criteria.

In short, the contributions of this study are described as follows:

- Propose an alternative solution for UPMSPs with JSD-MDSTs based on a modified WOA.
- The FA is applied as a local search for WOA to enhance its search ability.
- The proposed method had been evaluated using a set of UPMSP benchmark problems and compared to several state-of-arts methods. The evaluation outcomes approved the high performance of the WOAFA.

The rest of this study is given as follows. The problem definition is presented in Sect. 2, where the proposed WOAFA is presented in Sect. 3. The experimental evaluation and comparison outcomes are given in Sect. 4. Section 5 presents the conclusion and future direction.

2 Preliminaries

This section describes the problem definition, the main concept of the whale optimization algorithm (WOA), and the firefly algorithm (FA).

2.1 Problem definition

The problem of UPMSP can be mathematically modeled as a mixed integer programming (MIP). The following equations define this model (Ezugwu and Akutsah 2018):

$$Min\ C_{max} \tag{1}$$

Subject to

$$\sum_{i=0, i\neq 1}^{N_J} \sum_{k=1}^{N_m} x_{ijk} = 1; \forall j = 1, ..., N_J \tag{2}$$

$$\sum_{i=0, i\neq h}^{N_J} x_{ihk} - \sum_{j=0, j\neq h}^{N_J} x_{hjk} = 0;\ \forall h = 1, ..., N_J, k = 1, ..., N_m \tag{3}$$

$$C_j \geq C_i + \sum_{k=1}^{N_m} x_{ijk}(S_{ijk} + p_{ik}) + V\left(\sum_{k=1}^{N_m} x_{ijk} - 1\right),\ i = 0, ..., N_J \tag{4}$$

$$\sum_{j=0}^{N_J} x_{0jk} = 1, \forall k = 1, ..., N_m \tag{5}$$

$$C_j \leq C_{max}, \forall j = 1, ..., N_J, \tag{6}$$

$$x_{ijk} \in \{0, 1\}, \forall i = 0, ..., N_J, \forall j = 1, ..., N_J,\ \forall k = 1, ..., N_m, \tag{7}$$

$$C_0 = 0 \tag{8}$$

$$C_j \geq 0,\ \forall j = 0, ..., N_J \tag{9}$$

where Eq. 1 works to minimize C_{max}, whereas C_{max} denotes the total time required to complete the given process (makespan).

Equation 2 works to ensure that each job is performed on only one machine. When the job jth is executed after the job ith on the machine kth, the $x_{i,j,k}$ value will equal to 1; else, the value will be 0. Also, the $x_{i,j,k}$ value will equal to 1 if the jth job is the last job on machine k. N_J and N_m are jobs' number and of machines number, respectively.

Equation 3 is applied to ensure that there is only one succeeding job and one preceding job. C_j in Eq. 4 is the completion time of jth job, whereas $S_{i,j,k}$ denotes the sequence-dependent setup time to execute the jobs jth and ith in sequence order on kth machine. The jth job can be the first job on machine kth if the i equals to 0 as $S_{0,j,k}$ $p_{j,k}$ denotes the computation time of the jth job on machine k; this constraint is used to ensure the order of the jobs. This can be performed using a large number ($V = \infty$), where $\sum_{k=1}^{N_m} x_{ijk} = 1$ if the job jth is ordered after the job ith, therefore, $V(\sum_{k=1}^{N_m} x_{ijk} - 1) = 0$ and $C_j = C_i + p_{jk} + S_{ijk}$. Else, when the job jth does not ordered after the job ith, then $\sum_{k=1}^{N_m} x_{ijk} = 0$, and thus, $V(\sum_{k=1}^{N_m} x_{ijk} - 1) = -V$.

In Eq. 5, $x_{0,j,k} = 1$ when the jth job is the first job on machine kth; else $x_{0,j,k} = 0$. This equation is employed to ensure that only one job is listed first at each machine.

Equation 6 checks the completion time C_j of any job to be less than the C_{max}, whereas the solution x takes a binary value by Eq. 7. Equation 8 gives zero completion time to job 0, while Eq. 9 ensures that all completion times for all jobs are larger than zero.

2.2 Whale optimization algorithm (WOA)

The WOA is an optimization method proposed by Mirjalili and Lewis (2016). It emulates the behavior of the humpback whales. In WOA, the whale position represents the solution of the problem and is updated based on the behavior of the whale in attacking the prey Z_b. There are two attacking methods; the first one is the encircling method. It updates the whale position using Eq. 10

$$D_i = |B \times Z_b(t) - Z_i(t)| \tag{10}$$

$$Z_i(t+1) = Z_b(t) - A \times D_i \tag{11}$$

where D_i denotes the distance between $Z_b(t)$ and $Z_i(t)$. t is the current iteration. A and B are coefficients and calculated by:

$$A = 2a \times r - a \tag{12}$$

$$B = 2r \tag{13}$$

$$a = a - t\frac{a}{t_m} \tag{14}$$

where r is a random value $\in [0, 1]$. a is constantly decreased from 2 to 0 in the updating phase. t_m is the max iterations number).

The second one is the bubble-net method. It applied the shrinking encircling (if a is decreased in Eq. 12) or spiral updating position by applying the helix-shaped to simulate the whale movement that can be applied by the following equation:

$$Z(t+1) = D' \times e^{bl} \times cos(2\pi l) + Z_b(t) \tag{15}$$

where b is a random variable. l is also a random number $\in [-1, 1]$

In addition, the whales can swim around the Z_b using the shrinking circle and a spiral-shaped path at the same time. Therefore, Eq. 16 can be applied to update the position:

$$Z(t+1) = \begin{cases} Z_b(t) - A \times D & if \quad p \geq 0.5 \\ D' \times e^{bl} \times cos(2\pi l) + Z_b(t) & if \quad p < 0.5 \end{cases} \tag{16}$$

The WOA can also update its position by using a random search whale Z_r instead of the best whale Z^* as in the following equation:

$$Z(t+1) = Z_r - A \times D \tag{17}$$

$$D = |B \times Z_r - Z(t)| \tag{18}$$

The steps of the WOA are described in Algorithm 1.

Algorithm 1 Whale optimization algorithm

```
Generate the whale's population Z with size N.
Calculate the fitness function each whale.
Determine the best whale based on the fitness values.
while t <= t_m do
  for each solution Z_i do
    Calculate the parameters a, p, A, and B.
    if p ≥ 0.5 then
      Update Z_i using Equation (15).
    else
      if | A |≥ 1 then
        Update Z_i using Equation (17).
      else
        Update Z_i using Equation (10).
      end if
    end if
  end for
  t =t+1
end while t < t_m
```

2.3 Firefly algorithm (FA)

The FA is an optimization algorithm proposed by Yang (2010). It is based on the simulation of the flashing behavior of fireflies in nature. The behaviors of the FA are based on the rule: as the firefly is unisex, each one attracts to other ones. Also, the brighter one will attract the less bright one. Therefore, there is a relation between the brightness and attractiveness, whereas the attractiveness and brightness are increased if the distance between fireflies is decreased. If any firefly is brighter than the others, it will fly randomly. In addition, the firefly's brightness is depended on the landscape of the objective function.

Moreover, the attractiveness (β) between two fireflies can be calculated by Eq. 19:

$$\beta = \beta_0 \times e^{(-\gamma m^2)} \tag{19}$$

where γ denotes the light absorption coefficient. $\beta_0 = 1$ is the attractiveness at the distance m between i-th and Z_i fireflies equals 0. The m is calculated by the following equation:

$$m_{ij} = ||Z_i - Z_j|| = \sqrt{\sum_{k=1}^{d}(Z_{ik} - Z_{jk})^2} \tag{20}$$

The i firefly moves to another brighter j firefly by the following equation:

$$Z_i = Z_i + \beta \times (Z_i - Z_j) + r_4 \times \varepsilon_i \tag{21}$$

where, r_4 is a random value $\in [0, 1]$. $\varepsilon_i \in N(\mu, \sigma)$ denotes a random vector.

Algorithm 2 Firefly algorithm

```
Generate the fireflies solutions Z_i (i = 1, 2, ..., N).
Compute the fitness value for Z_i.
Light intensity I_i at Z_i is determined by f(Z_i)
Determine the light absorption coefficient γ
repeat
  for i = 1 : N do
    for j = 1 : i do
      if f_i < f_j then
        Move the i-th firefly Z_i towards the j-th firefly.
      end if
      Update the attractiveness.
      Update the position of firefly.
    end for
  end for
  Save the best solution.
until ( Stop condition )
Output the best solution.
```

3 The proposed WOAFA

This section introduces the proposed WOAFA to solve UPMSP with setup times. It utilizes both WOA and FA algorithms. In more detail, the operators of FA are utilized to enhance the exploitation ability of the WOA by serving as a local search. Figure 1 illustrates the flowchart of the proposed WOAFA.

In general, the WOAFA begins by creating a random integer population which represents the solution to start solving the UPMSP. Then, the WOA initials optimizing the solutions X. The solutions are evaluated using the objective function (to minimize the makespan). Therefore, the smallest makespan indicates the best solution. After that, each solution is updated using either the operators of WOA or the operators of FA. This switching is performed using a condition based on the probability. This condition is calculated by Eq. 22. All optimizing steps are repeated until reaching the stop condition. In the following subsections, the proposed WOAFA is described in detail.

3.1 Initial solution

The WOAFA begins by initializing the values of all parameters. Then it creates a random population X. The dimension of this population is the number of jobs N_J in the interval $[1, N_m]$. For instance, assume there are 10 jobs and 4 machines; therefore, the population X can be represented as $[x_1, x_2, x_3, ..., x_{N_J}] = [3\ 1\ 4\ 3\ 2\ 3\ 1\ 1\ 3\ 2]$. As in this example, the jobs (1, 4, 6, 9) will be implemented on machine number three, whereas jobs (2, 7, 8) will be implemented on machine number one, while jobs (3) and jobs (5, 10) will be implemented on machine number four and two, respectively. After that, the fitness function is computed for the population X by Eq. 1 to determine the best solution.

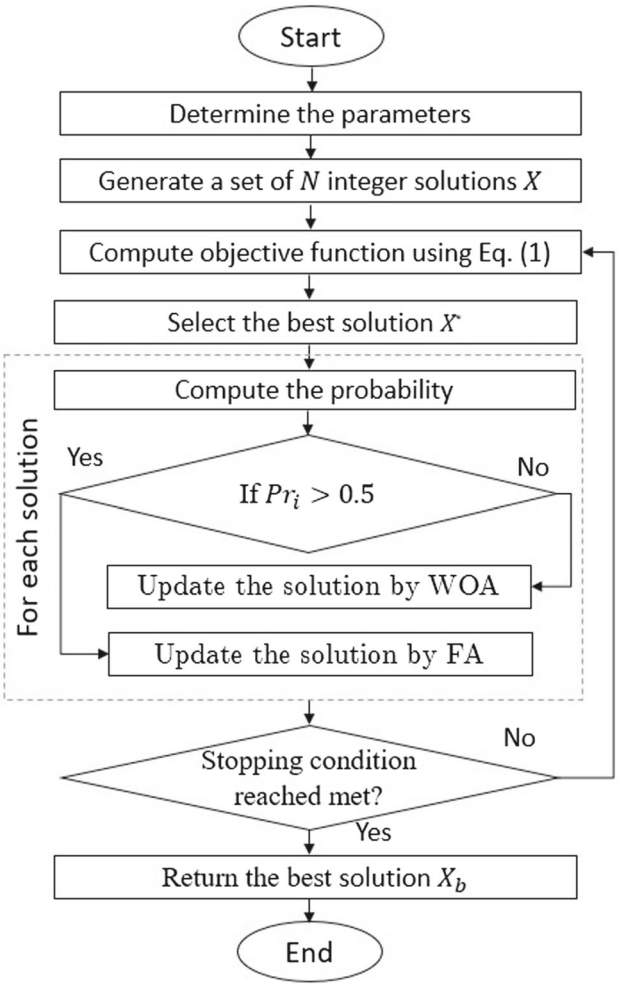

Fig. 1 Phases of the proposed method

3.2 Updating solution

To update the population, the proposed WOAFA evaluates each solution $(Z_i, i = 1, 2, ..., N)$ to check its quality by applying the fitness function as in Eq. 1 and the solution with the smallest select C_{max} is determined as the best solution Z_b. Then, the WOAFA calculates the probability (Pr_i) for each individual as in Eq. 22:

$$Pr_i = \frac{Fit_i}{\sum_{k=1}^{N} Fit_k}, \tag{22}$$

where Fit_i is the fitness value for the current solution and Fit_K is the total of the fitness values. Thence, if $Pr_i > 0.5$, then the FA will be applied to update the solution Z_i; else, the WOA will be applied.

The stop condition is checked after each loop; if it is true, the WOAFA will terminate the process ad output the best solution; else, the WOAFA will repeat its steps. The steps of the WOAFA are given in Algorithm 3.

Algorithm 3 The steps of the WOAFA

```
Initial the values for all variables and the initial solutions with dimension N_J.
Evaluate the quality of each Z_i by computing its fitness value using 1.
Save the best solution Z_b.
repeat
  for i = 1 : N do
    Compute the probability Pr_i
    if Pr_i > 0.5 then
      Update the solution using the operators of FA.
    else
      Update the solution using the operators of WOA.
    end if
    Update the parameters.
    Evaluate the quality of each Z_i by the fitness function.
    Save the best solution Z_b.
  end for
until Stop condition
```

The complexity of the developed WOAFA depends on the number of iterations, size of the population, and the dimension of the tested problem. In addition, it depends on the complexity of the quick sort algorithm. Therefore, the complexity of WOAFA can be formulated as:

$$O(WOAFA) = N_{Prob} * O(WOA) + (N - N_{Prob}) * O(FA) \quad (23)$$

So, in the best case,

$$O(WOAFA) = O(t \times (N_{Prob} \times N \times D + (N - N_{Prob}) \times (N \times D + NlogN))) \quad (24)$$

$$O(WOAFA) = O(t \times N(N \times D + (N - N_{Prob}) \times logN)), \quad (25)$$

while, in the worst case,

$$O(WOAFA) = O(t \times N^2(D + N - N_{Prob})) \quad (26)$$

where N_{Prob} is the number of solutions updated using operators of WOA.

4 Evaluation experiments and discussions

In this section, the WOAFA is evaluated for solving UPMSP using a benchmark dataset. Different numbers of jobs and machines are used. The results of the WOAFA are evaluated and compared with well-known meta-heuristic optimization methods, namely SA, PSO, GA, FA, ABC, ACO, WOA, and SCA.

The dataset used in this study is well-known dataset (WebSite 2019 (accessed Oct. 1, 2019).

It contains six types of jobs N_J (i.e., 20, 40, 60, 80, 100, and 120) and six types of jobs machines N_m (i.e., 2, 4, 6, 8, 10, and 12) in a discrete uniform distribution U[50, 100]. We evaluate 36 problems; each problem ($N_J \times N_m$) has 15 replication instances; thence, the total number of running is 540 times.

The results in the experiments are presented by the relative percentage deviation (RPD). This measure is used to compare the results and determine the performance of the proposed method against other methods. RPD is calculated by Eq. 27.

$$RPD = \frac{C_{max}(method) - C_{max}(WOAFA)}{C_{max}(WOAFA)} \times 100, \quad (27)$$

where $C_{max}(method)$ denotes the mean of the makespan values of the each method.

The experiments were performed using Core i5 CPU and 8 GB RAM under MS-Windows 10 (64bit) using MATLAB R2014b. The average results of 15 different instances were computed for each problem. For a fair comparison, all methods were applied in the same environment; besides, the stop condition in this study was set to the max computation time in milliseconds. Therefore, the optimization process runs for a part of the time proportional to the job numbers and machine numbers.

$$maxTime = n(\frac{m}{2}) \times tc \quad (28)$$

where n and m are the numbers of current jobs and current machines, respectively. tc is set to 50 as recommended by Diana et al. (2015). The experiments' parameters for all methods are set as in their original papers. Besides, several previous papers are revised to determine the best paper meters for the proposed method such as (Mirjalili and Lewis 2016; Yang 2009; AbdElaziz et al. 2021; Alameer et al. 2019; Abd ElAziz et al. 2018; Mohamed et al. 2016).

4.1 Computational results

The comparison results between the WOAFA and other algorithms are illustrated in Tables 1-2 that show the average of RPD and standard deviation, respectively. From the results of RPD, it can notice that the WOAFA outperforms other algorithms overall the tested instances except at the machines 2 and Jobs from 80:120, the FA is better. By analyzing the results of RPD at each job, it can be observed that there are variants of RPD values: some of them are under 25% and this includes SA, GA, FA, ACO, WOA, and SCA. The second category has RPD higher than 25%, and this contains three algorithms, namely PSO, GA, and ABC, as shown in Fig. 2.

Moreover, Figure 3 shows the average of RPD at each machine, and it can be seen that the WOAFA method provides

Table 1 Results of the relative percentage deviation (RPD) for all methods

Machines	Jobs	SA	PSO	GA	FA	ABC	ACO	WOA	SCA
2	20	0.29	16.45	16.45	1.93	45.84	34.85	0.53	0.32
	40	2.27	15.95	15.91	0.18	13.02	4.05	0.59	0.09
	60	4.05	18.47	18.36	0.27	14.68	6.40	1.53	0.47
	80	4.79	17.11	16.83	**-1.35**	13.54	5.48	1.27	0.04
	100	60.00	18.63	18.57	**-1.21**	14.68	8.02	1.93	0.13
	120	7.12	18.95	18.91	**-1.59**	16.43	9.74	1.72	0.02
4	20	18.92	63.12	64.35	18.48	39.25	18.32	11.21	5.50
	40	5.48	50.33	51.31	1.61	28.94	9.51	18.37	6.81
	60	10.37	51.29	52.14	6.20	37.45	13.60	10.21	2.16
	80	12.87	63.29	62.18	6.37	36.08	15.00	11.35	1.20
	100	14.92	58.02	57.61	9.16	35.23	18.19	8.36	3.34
	120	5.90	51.98	49.38	0.01	23.48	9.07	9.47	6.24
6	20	9.28	61.70	61.70	9.51	25.68	9.80	8.06	5.20
	40	5.90	88.31	88.31	2.72	38.18	14.00	19.04	15.49
	60	5.61	77.64	77.83	0.32	44.19	9.56	5.25	2.41
	80	8.44	85.88	85.87	1.03	46.10	14.09	5.59	6.42
	100	18.27	73.23	74.04	0.49	41.22	14.16	20.10	16.02
	120	14.55	90.03	87.85	1.96	47.62	14.94	6.79	1.00
8	20	19.70	46.58	46.58	20.36	27.61	22.40	1.22	17.90
	40	6.63	76.03	76.03	6.74	41.21	7.31	6.06	13.19
	60	6.18	118.36	118.36	3.62	53.33	9.62	8.35	5.39
	80	24.28	97.24	94.48	1.79	46.65	12.10	14.07	6.61
	100	11.30	105.39	105.59	0.01	57.85	24.37	13.74	4.02
	120	13.94	110.39	109.57	2.64	50.31	20.56	14.74	5.17
10	20	32.11	59.92	59.92	3.54	34.91	32.11	34.29	1.86
	40	16.29	48.98	48.98	15.65	25.99	16.52	9.51	5.67
	60	5.28	88.95	88.95	1.96	36.83	9.49	17.76	7.65
	80	9.28	112.65	113.45	0.56	53.98	10.26	8.31	7.66
	100	16.38	128.10	128.75	0.03	56.29	20.65	11.07	4.96
	120	21.94	128.66	129.29	5.09	69.09	27.92	19.75	7.83
12	20	18.77	33.66	33.66	18.60	19.54	18.90	15.10	7.37
	40	14.70	57.46	57.50	14.32	25.90	16.37	2.74	6.84
	60	22.29	96.45	96.37	8.65	43.86	20.78	14.87	37.30
	80	14.11	109.02	109.02	9.23	43.36	17.23	15.78	19.44
	100	26.38	135.90	136.10	11.81	79.15	35.98	22.45	27.42
	120	20.56	125.62	124.32	8.05	69.40	30.03	21.14	14.25

better makespan overall tested machines except at machine 2 where the FA provides better results WOAFA. Moreover, the average over all the tested machine and jobs that given in Fig. 4 illustrates the following; the large improvements are achieved at PSO, GA, and ABC, while the smallest improvement can be noticed by comparing the proposed WOAFA and the FA.

Furthermore, the results of STD are shown in Table 2 and Fig. 5. We can notice that the proposed WOAFA is more stable than others, followed by FA and SA, that allocate the second and third rank, respectively. Also, PSO and GA are the worst stable methods. This high performance of the developed WOAFA method results from combining the advantages of WOA and FA. This combination makes the WOA and FA to update the solutions in competitive manner according to the quality of each solution.

In addition, the computational times for all methods are recorded in Table 3. From the table, we can see that the WOAFA method showed competitive results compared to the other method, especially with SA, PSO, and WOA. The WOAFA improved the computational time of the original FA because it has an effective ability to balancing between the

Table 2 Results of the STD for all methods

Machines	Jobs	SA	PSO	GA	FA	ABC	ACO	WOA	SCA	WOAFA
2	20	**28.10**	43.92	43.92	39.74	610.37	597.99	51.75	29.58	37.82
	40	33.27	43.32	41.97	13.70	71.35	38.41	78.53	**8.73**	20.95
	60	74.98	89.78	87.13	**27.31**	82.64	67.79	131.57	55.84	33.65
	80	64.14	71.04	64.50	46.31	74.43	67.29	93.62	27.52	**24.18**
	100	48.67	111.99	108.24	26.37	79.75	62.80	111.07	**25.76**	29.14
	120	98.94	100.49	89.74	27.60	168.38	77.68	113.94	**18.38**	20.71
4	20	182.38	181.68	176.90	186.45	173.33	185.60	94.04	66.72	**26.46**
	40	51.21	281.93	271.21	36.20	105.92	110.28	171.02	119.35	**8.39**
	60	223.97	395.52	403.15	245.18	292.21	231.50	178.21	**23.61**	29.17
	80	265.44	607.16	566.45	296.58	247.56	277.41	304.66	52.90	**19.14**
	100	502.15	822.35	795.31	452.37	667.53	508.89	280.54	274.93	**25.00**
	120	40.56	841.74	684.39	38.53	240.77	73.22	373.24	405.06	**31.48**
6	20	64.64	167.06	167.06	65.90	79.98	74.82	103.74	108.08	**58.42**
	40	76.83	341.38	341.38	77.55	109.18	81.30	129.33	154.73	**66.03**
	60	166.27	430.98	427.94	163.54	262.52	171.61	120.17	141.45	**78.77**
	80	126.08	493.83	470.44	106.37	280.64	218.58	120.77	120.79	**78.38**
	100	598.11	500.47	492.55	121.41	323.54	177.63	330.16	353.22	**86.00**
	120	301.61	636.02	607.61	188.43	514.58	263.07	198.17	129.74	**103.92**
8	20	156.82	**119.88**	**119.88**	155.68	139.60	155.06	135.63	152.21	143.47
	40	152.01	219.39	219.39	150.53	**69.87**	148.96	92.66	178.49	95.23
	60	110.51	524.74	524.74	110.71	222.17	103.84	107.48	88.94	**67.38**
	80	460.81	280.55	240.73	201.00	374.42	215.39	244.55	119.32	**108.48**
	100	125.82	611.19	606.10	129.94	332.05	179.72	144.05	133.88	**124.46**
	120	146.69	476.00	455.39	158.47	384.00	275.06	194.57	133.60	**109.79**
10	20	187.38	178.62	178.62	77.77	175.48	187.38	263.23	**52.61**	89.40
	40	273.43	**110.50**	**110.50**	280.22	190.90	271.01	285.87	166.06	336.92
	60	166.55	224.34	224.34	185.38	**82.13**	174.71	143.89	194.02	90.05
	80	165.74	440.42	432.99	90.49	189.81	100.16	156.43	106.04	**82.80**
	100	126.07	603.51	600.76	**79.05**	146.44	168.29	103.89	114.35	101.91
	120	382.61	680.49	670.43	154.24	239.56	210.94	163.79	95.62	**39.39**
12	20	293.46	245.85	245.85	294.47	289.15	292.69	249.13	266.48	**187.39**
	40	359.06	**237.44**	237.56	360.08	296.62	351.93	295.61	329.68	304.32
	60	154.93	268.12	268.29	140.96	126.59	159.79	198.03	348.95	**120.48**
	80	202.82	394.99	394.99	240.49	191.85	**161.08**	182.37	261.69	169.63
	100	140.65	352.89	352.47	133.57	186.99	220.49	113.45	172.75	**109.20**
	120	52.74	332.86	325.05	134.39	198.02	147.14	132.93	60.01	**46.48**

two algorithms to obtain the optimal results in a short time. The average CPU time for all machines and jobs is illustrated in Fig. 6.

4.2 Statistical analysis

This subsection shows the results of the Wilcoxon-rank sum test as a non-parametric test. It is employed to indicate if there are significant differences between the WOAFA and the compared methods or not at the value of the $p - value$ less than 0.05. Table 4 lists the test results, and we can notice the significant differences between the proposed method (WOAFA) and PSO, GA, and ABC. Whereas other methods obtained the $p - value$ greater than 0.05, this can be because the results' difference of these methods is small in the same computation time; however, the WOAFA obtained the best results in most of all problems.

Furthermore, the Friedman test is used to statistically rank the algorithms, where the lowest Friedman's value refers to the best algorithm; Table 5 lists the results for each machine. From the table, we can see that the WOAFA method obtained the best rank in all machines, and therefore, it achieved the

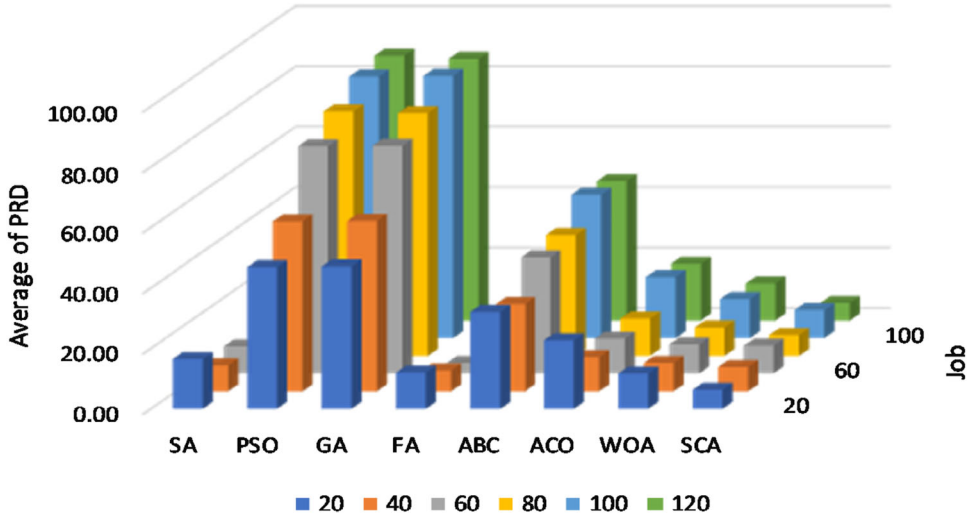

Fig. 2 Average of RPD at each job

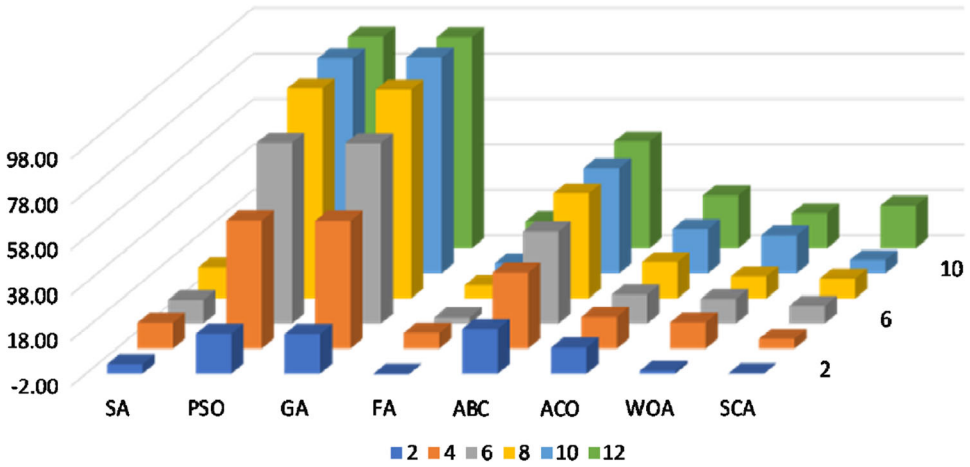

Fig. 3 Average of RPD at each machine

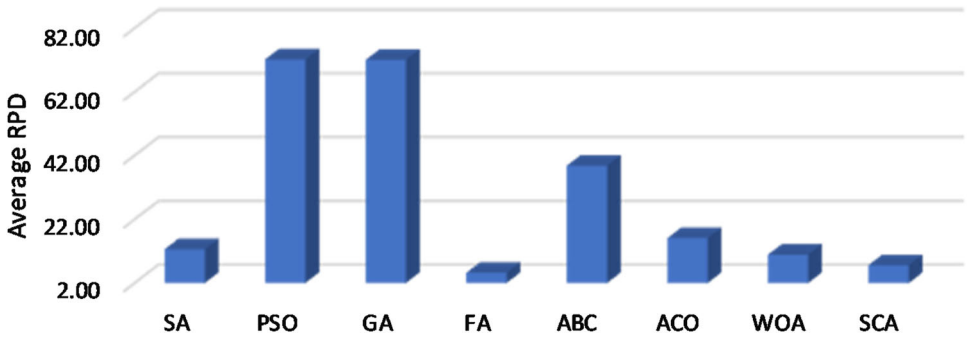

Fig. 4 Average of PRD overall the tested machine and job

first rank. The FA obtained the second rank in all machines except for machine 2, followed by the SCA, WOA, SA, ACO, ABC, GA, and PSO, respectively.

To sum up the above discussions, it can be noticed that the best results of the WOAFA because the WOAFA befits from the advantages of both WOA and FA such as the WOA have many stages and strategies to update the solutions and explore the search domain, whereas the FA helps the proposed method to increase its exploitation and exploration phases. In contrast, the WOAFA has some limitations, such as it falls in local optima in the machine (2) at jobs (80, 100, and 120); therefore, it needs to be improved in solving the small number of machines. The proposed method's stability is good, but it needs more enhancement, especially in machines (2 and 10).

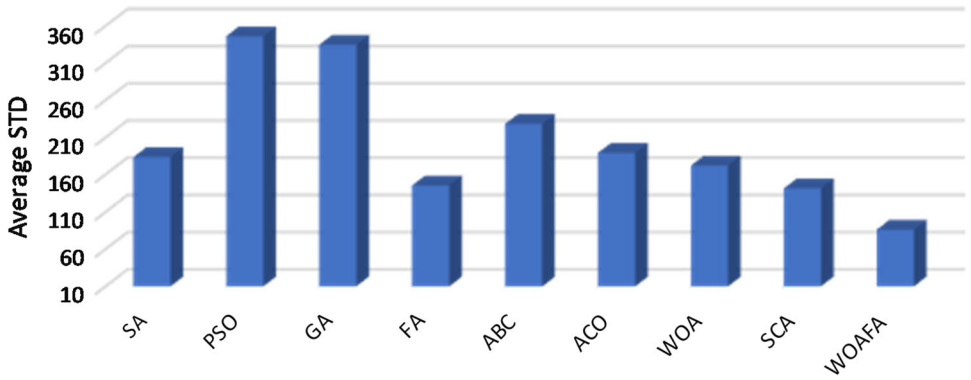

Fig. 5 Average of STD overall the tested machine and job

Table 3 Results of the computation time for all methods

Machines	Jobs	SA	PSO	GA	FA	ABC	ACO	WOA	SCA	WOAFA
2	20	1.002	1.002	1.007	1.005	1.007	1.002	1.001	1.006	1.006
	40	2.004	2.003	2.006	2.006	2.013	2.009	2.002	2.007	2.009
	60	3.002	3.005	3.011	3.009	3.014	3.046	3.003	3.008	3.003
	80	4.005	4.010	4.019	4.017	4.020	4.074	4.006	4.014	4.009
	100	5.003	5.006	5.018	5.020	5.029	5.156	5.011	5.014	5.015
	120	6.003	6.017	6.026	6.583	6.018	6.065	6.011	6.031	6.011
4	20	2.001	2.002	2.003	2.003	2.004	2.006	2.001	2.004	2.004
	40	4.001	4.001	4.006	4.006	4.013	4.023	4.002	4.007	4.007
	60	6.002	6.001	6.012	6.009	6.020	6.057	6.005	6.007	6.010
	80	8.002	8.002	8.021	8.014	8.031	8.075	8.005	8.013	8.014
	100	10.003	10.006	10.026	10.015	10.021	10.033	10.012	10.019	10.016
	120	12.003	12.019	12.037	12.025	12.038	12.260	12.006	12.023	12.023
6	20	3.001	3.002	3.005	3.004	3.008	3.002	3.003	3.003	3.004
	40	6.002	6.003	6.010	6.006	6.006	6.013	6.003	6.007	6.008
	60	9.001	9.003	9.014	9.009	9.023	9.056	9.004	9.009	9.011
	80	12.002	12.002	12.022	12.016	12.028	12.031	12.002	12.016	12.012
	100	15.002	15.005	15.015	15.019	15.023	15.024	15.013	15.017	15.023
	120	18.002	18.010	18.022	18.027	18.052	18.033	18.009	18.026	18.033
8	20	4.001	4.001	4.004	4.003	4.004	4.005	4.002	4.004	4.005
	40	8.001	8.004	8.007	8.005	8.013	8.011	8.004	8.005	8.007
	60	12.002	12.005	12.016	12.011	12.017	12.060	12.005	12.011	12.013
	80	16.002	16.003	16.012	16.014	16.034	16.049	16.004	16.020	16.015
	100	20.003	20.013	20.032	20.021	20.049	20.210	20.013	20.021	20.020
	120	24.003	24.013	24.026	24.021	24.046	24.104	24.007	24.023	24.022
10	20	5.001	5.002	5.004	5.004	5.008	5.004	5.003	5.005	5.004
	40	10.001	10.004	10.011	10.008	10.016	10.008	10.005	10.006	10.009
	60	15.001	15.004	15.013	15.013	15.011	15.015	15.004	15.011	15.010
	80	20.002	20.009	20.021	20.013	20.023	20.056	20.005	20.019	20.015
	100	25.004	25.008	25.024	25.023	25.020	25.124	25.012	25.021	25.024
	120	30.004	30.005	30.029	30.026	30.050	30.212	30.003	30.024	30.023
12	20	6.001	6.002	6.005	6.006	6.004	6.001	6.001	6.005	6.004
	40	12.001	12.002	12.007	12.007	12.006	12.018	12.001	12.010	12.007
	60	18.002	18.006	18.018	18.013	18.025	18.010	18.007	18.013	18.013
	80	24.002	24.011	24.019	24.018	24.028	24.102	24.011	24.016	24.020
	100	30.003	30.016	30.032	30.019	30.034	30.070	30.013	30.021	30.013
	120	36.004	36.006	36.033	36.028	36.050	36.321	36.015	36.020	36.015

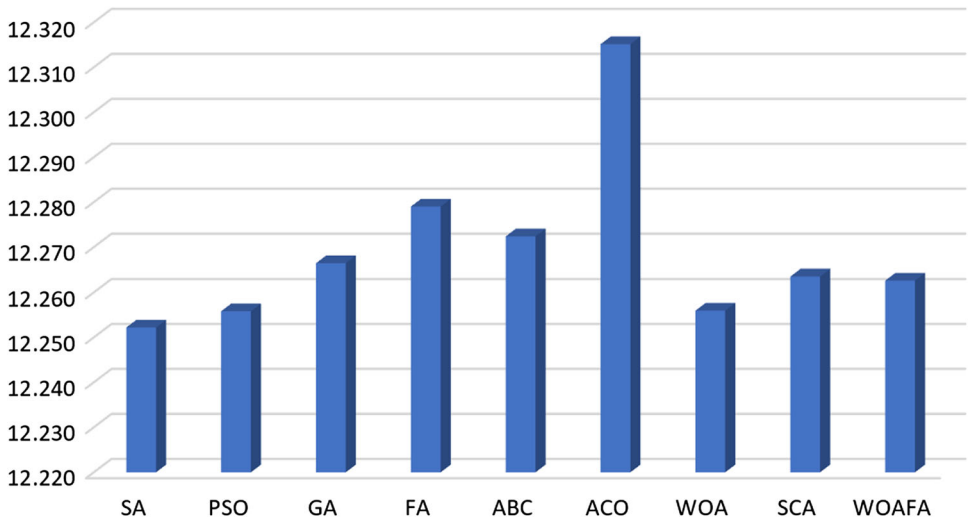

Fig. 6 Average of the computation time (in seconds) overall the tested machines and jobs

Table 4 Results of Wilcoxon rank-sum test

	SA	PSO	GA	FA	ABC	ACO	WOA	SCA
p-value	0.440	0.003	0.003	0.689	0.044	0.359	0.427	0.570

Table 5 Results of Friedman rank

Machine	SA	PSO	GA	FA	ABC	ACO	WOA	SCA	WOAFA
2	4.500	8.583	7.750	2.167	7.333	6.333	4.000	2.833	1.500
4	4.500	8.500	8.500	3.167	7.000	5.333	4.333	2.667	1.000
6	4.500	8.500	8.500	2.667	7.000	5.167	4.333	3.333	1.000
8	4.167	8.583	8.417	2.833	7.000	5.500	4.000	3.500	1.000
10	4.583	8.250	8.750	2.500	7.000	5.583	4.500	2.833	1.000
12	4.333	8.500	8.500	2.667	7.000	5.500	3.333	4.167	1.000

5 Conclusions

This study proposed an alternative meta-heuristic-based solution for unrelated parallel machine scheduling problems (UPMSPs) with job sequence-dependent and machine-dependent setup times (JSDMDSTs). The proposed WOAFA method is a hybrid of whale optimization algorithm (WOA) and firefly algorithm (FA). The FA is employed as a local search method to enhance the exploitation ability of the WOA. We used a well-known benchmark dataset to test the performance of the proposed WOAFA, including six machines (i.e., 2, 4, 6, 8, 10, and 12 machines) and six jobs (i.e., 20, 40, 60, 80, 100, and 120 jobs). Moreover, we compare the proposed WOAFA with several meta-heuristics, such as FA, WOA, SA, PSO, GA, ACO, ABC, and SCA. Furthermore, the Wilcoxon rank-sum test was employed to evaluate the proposed method over other methods. Overall results showed that the proposed WOAFA outperforms several previous methods.

According to the good performance of the proposed WOAFA, in the future, it may be applied in other optimization problems, such as feature selection, scheduling issues of cloud computing, image processing, or time series forecasting.

Declarations

Conflict of interest All authors declare that they have no conflict of interest

Human and animal rights This article does not contain any studies with human participants or animals performed by any of the authors.

References

Abd ElAziz M, Ewees AA, Hassanien AE (2018) Multi-objective whale optimization algorithm for content-based image retrieval. Multimed Tools Appl 77(19):26135–26172

AbdElaziz M, Nabil N, Moghdani R, Ewees AA, Cuevas E, Lu S (2021) Multilevel thresholding image segmentation based on improved volleyball premier league algorithm using whale optimization algorithm. Multimed Tools Appl. https://doi.org/10.1007/s11042-020-10313-w

Abdel-Basset M, Manogaran G, El-Shahat D, Mirjalili S (2018) A hybrid whale optimization algorithm based on local search strategy for the permutation flow shop scheduling problem. Fut Gen Comput Syst 85:129–145

Afzalirad M, Rezaeian J (2016) Resource-constrained unrelated parallel machine scheduling problem with sequence dependent setup times, precedence constraints and machine eligibility restrictions. Comput Ind Eng 98:40–52

Akyol S, Alatas B (2020) Sentiment classification within online social media using whale optimization algorithm and social impact theory based optimization. Physica A Stat Mech Appl 540:123094

Alameer Z, Abd Elaziz M, Ewees AA, Ye H, Jianhua Z (2019) Forecasting gold price fluctuations using improved multilayer perceptron neural network and whale optimization algorithm. Res Pol 61:250–260

Arroyo JEC, Leung JYT, Tavares RG (2019) An iterated greedy algorithm for total flow time minimization in unrelated parallel batch machines with unequal job release times. Eng Appl Artif Intell 77:239–254

Attiya I, Abd Elaziz M (2020) Xiong S (2020) Job scheduling in cloud computing using a modified harris hawks optimization and simulated annealing algorithm. Comput Intell Neurosci

Bektur G, Saraç T (2019) A mathematical model and heuristic algorithms for an unrelated parallel machine scheduling problem with sequence-dependent setup times, machine eligibility restrictions and a common server. Comput Oper Res 103:46–63

De Paula MR, Ravetti MG, Mateus GR, Pardalos PM (2007) Solving parallel machines scheduling problems with sequence-dependent setup times using variable neighbourhood search. IMA J Manag Math 18(2):101–115

Diana ROM, de França Filho MF, de Souza SR, de Almeida Vitor JF (2015) An immune-inspired algorithm for an unrelated parallel machines' scheduling problem with sequence and machine dependent setup-times for makespan minimisation. Neurocomputing 163:94–105

El Aziz MA, Ewees AA, Hassanien AE (2017) Whale optimization algorithm and moth-flame optimization for multilevel thresholding image segmentation. Expert Syst Appl 83:242–256

Ezugwu AE, Akutsah F (2018) An improved firefly algorithm for the unrelated parallel machines scheduling problem with sequence-dependent setup times. IEEE Access 6:54459–54478

Fanjul-Peyro L, Ruiz R, Perea F (2019) Reformulations and an exact algorithm for unrelated parallel machine scheduling problems with setup times. Comput Oper Res 101:173–182

Hamzadayi A, Yildiz G (2016a) Event driven strategy based complete rescheduling approaches for dynamic m identical parallel machines scheduling problem with a common server. Comput Indus Eng 91:66–84

Hamzadayi A, Yildiz G (2016b) Hybrid strategy based complete rescheduling approaches for dynamic m identical parallel machines scheduling problem with a common server. Simul Modell Pract Theory 63:104–132

Hamzadayi A, Yildiz G (2017) Modeling and solving static m identical parallel machines scheduling problem with a common server and sequence dependent setup times. Comput Indus Eng 106:287–298

Helal M, Rabadi G, Al-Salem A (2006) A tabu search algorithm to minimize the makespan for the unrelated parallel machines scheduling problem with setup times. Int J Oper Res 3(3):182–192

Jain L, Katarya R (2019) Discover opinion leader in online social network using firefly algorithm. Expert Syst Appl 122:1–15

Jouhari H, Lei D, Alqaness AA, Abd Elaziz M, Ewees AA, Farouk O (2019) Sine-cosine algorithm to enhance simulated annealing for unrelated parallel machine scheduling with setup times. Mathematics 7(11):1120

Kim MY, Lee YH (2012) Mip models and hybrid algorithm for minimizing the makespan of parallel machines scheduling problem with a single server. Comput Oper Res 39(11):2457–2468

Lin SW, Ying KC (2014) Abc-based manufacturing scheduling for unrelated parallel machines with machine-dependent and job sequence-dependent setup times. Comput Oper Res 51:172–181

Lin SW, Lu CC, Ying KC (2011) Minimization of total tardiness on unrelated parallel machines with sequence-and machine-dependent setup times under due date constraints. Int J Adv Manufact Technol 53(1–4):353–361

Logendran R, McDonell B, Smucker B (2007) Scheduling unrelated parallel machines with sequence-dependent setups. Comput Oper Res 34(11):3420–3438

Mafarja MM, Mirjalili S (2017) Hybrid whale optimization algorithm with simulated annealing for feature selection. Neurocomputing 260:302–312

McNaughton R (1959) Scheduling with deadlines and loss functions. Manag Sci 6(1):1–12

Mir MSS, Rezaeian J (2016) A robust hybrid approach based on particle swarm optimization and genetic algorithm to minimize the total machine load on unrelated parallel machines. Appl Soft Comput 41:488–504

Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67

Mirjalili S, Mirjalili SM, Saremi S, Mirjalili S (2020) Whale optimization algorithm: theory, literature review, and application in designing photonic crystal filters. In: Nature-Inspired Optimizers, Springer, pp 219–238

Mohamed A, Ewees AA, Hassanien AE (2016) Hybrid swarms optimization based image segmentation. In: Hybrid soft computing for image segmentation, Springer, pp 1–21

Nayak J, Vakula K, Dinesh P, Naik B (2020) Applications and advancements of firefly algorithm in classification: An analytical perspective. In: Computational Intelligence in Pattern Recognition, Springer, pp 1011–1028

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

Pakzad-Moghaddam S (2016) A lévy flight embedded particle swarm optimization for multi-objective parallel-machine scheduling with learning and adapting considerations. Comput Indus Eng 91:109–128

Qiao W, Yang Z, Kang Z, Pan Z (2020) Short-term natural gas consumption prediction based on volterra adaptive filter and improved whale optimization algorithm. Eng Appl Artif Intell 87:103323

Rajagopalan A, Modale DR, Senthilkumar R (2020) Optimal scheduling of tasks in cloud computing using hybrid firefly-genetic algorithm. Image Processing Security and Computer Vision. Advances in Decision Sciences. Springer, Newyork, pp 678–687

Rocha PL, Ravetti MG, Mateus GR, Pardalos PM (2008) Exact algorithms for a scheduling problem with unrelated parallel machines and sequence and machine-dependent setup times. Comput Oper Res 35(4):1250–1264

Santos HG, Toffolo TA, Silva CL, Vanden Berghe G (2019) Analysis of stochastic local search methods for the unrelated parallel machine scheduling problem. Int Trans Oper Res 26(2):707–724

Trivedi IN, Pradeep J, Narottam J, Arvind K, Dilip L (2016) Novel adaptive whale optimization algorithm for global optimization. Ind J Sci Technol 9(38):319–326

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(3):612–622

WebSite D (2019 (accessed Oct. 1, 2019)) Scheduling Research Dataset. http://www.schedulingresearch.com

Yang XS, He X (2013) Firefly algorithm: recent advances and applications. arXiv preprint arXiv:1308.3898
Yang XS (2009) Firefly algorithms for multimodal optimization. In: International symposium on stochastic algorithms, Springer, pp 169–178
Yang XS (2010) Nature-Inspired Metaheuristic Algorithms
Yang XS (2020) Firefly algorithm and its variants in digital image processing. Case studies and new developments, applications of firefly algorithm and its variants
Yilmaz Eroglu D, Ozmutlu HC, Ozmutlu S (2014) Genetic algorithm with local search for the unrelated parallel machine scheduling problem with sequence-dependent set-up times. Int J Prod Res 52(19):5841–5856
Ying KC, Lee ZJ, Lin SW (2012) Makespan minimization for scheduling unrelated parallel machines with setup times. J Intell Manuf 23(5):1795–1803
Zhou J, Nekouie A, Arslan CA, Pham BT, Hasanipanah M (2019) Novel approach for forecasting the blast-induced aop using a hybrid fuzzy system and firefly algorithm. Engineering with Computers pp 1–10