



Research Article

Uniform Parallel Machines Scheduling with Setup Time, Learning Effect, Machine Idle Time, and Processing Set Restrictions to Minimize Earliness/Tardiness Costs

Javad Rezaeian^{*} , Keyvan Shokoufi, Reza Alizadeh Foroutan 

Department of Industrial Engineering, Mazandaran University of Science and Technology, Babol, Iran
Email: j.rezaeian@ustmb.ac.ir

Received: 16 June 2021; **Revised:** 01 November 2021; **Accepted:** 04 November 2021

Abstract: Inspired by a real industrial case, this study deals with the problem of scheduling jobs on uniform parallel machines with past-sequence-dependent setup times to minimize the total earliness and tardiness costs. The paper contributes to the existing literature of uniform parallel machines problems by the novel idea of considering position-based learning effects along with processing set restrictions. The presented problem is formulated as a Mixed Integer linear programming (MILP) model. Then, an exact method is introduced to calculate the accurate objective function in the just-in-time (JIT) environments for a given sequence of jobs. Furthermore, three meta-heuristic approaches, (1) a genetic algorithm (GA), (2) a simulated annealing algorithm (SA), and (3) a particle swarm optimization algorithm (PSO) are proposed to solve large size problems in reasonable computational time. Finally, computational results of the proposed meta-heuristic algorithms are evaluated through extensive experiments and tested using ANOVA followed by *t*-tests to identify the most effective meta-heuristic.

Keywords: earliness-tardiness scheduling, uniform parallel machine, heuristic algorithm, setup time, learning effect, processing set restrictions

1. Introduction

Production scheduling is one of the main sectors of supply chain management and has always been one of the most important issues for manufacturers, especially in the Just-In-Time (JIT) production system [1]. Considering setup time usually happens in industrial settings when various types of jobs are processed on machines. There are two types of setup time: sequence-independent and sequence-dependent. In the first type, the setup time is usually added to the job processing time while in the second type, the setup time depends not only on the job currently being scheduled but also on the last scheduled job. Allahverdi et al. provided an extensive review of scheduling problems with setup times, including the parallel machines cases [2]. Considering sequence-dependent setup times, Ying et al. investigated unrelated machine scheduling problems. They presented a restricted simulated annealing (RSA) algorithm to minimize makespan [3]. Rezaeian et al. dealt with unrelated parallel machine scheduling problems with sequence-dependent setup times under a fully fuzzy environment to minimize total weighted fuzzy earliness and tardiness penalties [4]. Ramezani et al. studied a no-wait scheduling problem in a flexible flow shop environment with uniform parallel machines considering anticipatory sequence-dependent setup times to minimize makespan and developed a hybrid meta-heuristic to tackle the problem [5]. Cota et al. investigated unrelated parallel machine scheduling problems with sequence-

dependent setup times to minimize makespan and total consumption of electricity [6]. They considered independent and non-preemptible jobs and employed a novel meta-heuristic algorithm, named multi-objective smart pool search meta-heuristic to find near-Pareto solutions. Ahmadizar et al. dealt with an unrelated parallel machine scheduling problem with machine eligibility to minimize the total earliness and tardiness [7]. Setup times are both sequence-dependent and machine-dependent. Vallada and Ruiz proposed a Mixed Integer Programming (MIP) model formulation and a genetic algorithm to minimize makespan on unrelated parallel machine scheduling problems with sequence-dependent setup times [8]. Rezaeian et al. studied unrelated parallel machine scheduling problems with sequence-dependent setup time to minimize the sum of weighted earliness and tardiness costs and developed a mathematical formulation and a Pareto-based algorithm for the regarded problem [9]. Kim and Lee addressed a uniform parallel dedicated machine scheduling problem with machine eligibility, job splitting, sequence-dependent setup times, and limited setup servers and proposed a heuristic algorithm to tackle the problem [10].

During the last two decades, the topic of learning effects which is very popular in scheduling problems has been occasionally studied together with the topic of setup time. Biskup prepared a comprehensive review of scheduling problems with learning effects [11]. He categorized the models in the literature into two diverse classes: (1) the position-based learning problems, and (2) the sum of processing-time-based learning problems. Azzouz et al. employed an adaptive genetic algorithm to solve a flexible job-shop problem with sequence-dependent setup times and the learning effects to minimize makespan [12]. Shokoufi et al. addressed uniform parallel machine scheduling problems with time-dependent learning effect, release date, and allowable preemption to minimize the total weighted of earliness and tardiness penalties [13]. Azadeh et al. studied stochastic flexible flow shop with sequence-dependent setup times, job deterioration, and learning effects [14]. They presented an integrated approach based on artificial neural network (ANN), genetic algorithm (GA), and computer simulation to minimize the total tardiness of jobs in the sequences. Kuo et al. considered the total absolute deviation of job completion times and the total load on all machines as scheduling measures [15]. They studied unrelated parallel machine problems with past-sequence-dependent setup time and learning effects; showed that the proposed problem remains polynomial solvable. Liao et al. dealt with a two-competing group scheduling problem on serial-batching machines to minimize the makespan with some specific considerations [16]. Setup times and truncated job-dependent learning effects are taken into account, some structural properties and a greedy algorithm were proposed.

In addition to the above, many manufacturing constraints are caused by situations where machines have different capabilities and proficiency levels when facing tasks to be processed. These constraints may be due to limited speed, and lack of specific constituents which only some of the machines are equipped with. Li et al. proposed a hybrid differential evolution (HDE) algorithm embedded with chaos theory and two local search algorithms to minimize the total tardiness of a parallel machine scheduling problem with different color families, sequence-dependent setup times, and machine eligibility restriction [17]. Huo and Leung considered a scheduling problem of parallel machines in which each job can only be scheduled on a subset of machines [18]. They proposed an improved algorithm to minimize the makespan. Gokhale and Mathirajan addressed parallel machines with sequence-dependent setup times, unequal release times, and machine eligibility restrictions to minimize total weighted flow time [19]. Perez-Gonzalez et al. modeled unrelated parallel machines with machine eligibility and sequence-dependent setup times to minimize the total tardiness [20]. They selected and adapted some existing heuristics and a metaheuristic from related problems, as well as proposed a set of heuristics with novel repair and improvement phases as solution approaches.

One of the important objective functions in the literature is to minimize the sum of earliness and tardiness (E/T). JIT scheduling has emerged as a response to the necessity of fulfilling each customer's order at their most desired time [21]. In a JIT scheduling environment, the objective is to complete each job as close to its due date as possible [22]. Many research papers on scheduling problems have been published with both earliness and tardiness penalties. Based on the study of the literature, and to the best of the authors' knowledge, the uniform parallel machines scheduling problem which considers scheduling problem which considers, position-based learning effect, and processing set restrictions have not yet been investigated. Hence, this problem is addressed in this study to minimize the sum of earliness and tardiness. Following the notation system introduced by Graham et al. [23], the proposed problem is denoted as $Q|M_j, S_{ij}, LE|\sum E_j + T_j$ that is known to be strongly NP-hard because the simpler case of the E/T problem is an NP-hard problem [24]. To obtain the exact solution of the problem, a mathematical programming model is proposed. Also, three meta-heuristic algorithms for solving large-sized problems are presented. Furthermore, a heuristic algorithm is proposed to compute

the objective function of a given sequence.

The rest of the study is organized as follows: In Section 2, the problem under study is defined. The proposed mathematical model is presented in Section 3. In Section 4, a heuristic algorithm and three meta-heuristic algorithms are proposed and used to solve the sample problems. The experimental results and evaluations are given in Section 5. Finally, Section 6 provides conclusions of the study and suggests some directions for future researches.

2. Problem definitions

The automobile brake system industry is a real example of the considered problem. The brake system is the most important safety measure for cars. Drivers can safely stop or slow down the vehicle by repetitively actuating the system. Most brake systems make use of the hydraulic principle to convert pedal movement into braking force. Master cylinder is a critically important component in the conversion of pedal movement into hydraulic pressure. Master cylinder assembly is composed of four main parts; a cylinder body, a fluid reservoir, a primary piston, and a secondary piston made of steel or aluminum alloys. In the manufacturing process of master cylinder assembly, a set of n jobs $J_j, j = 1, \dots, n$ are available at time zero. Pistons are processed by a set of similar machines $M = \{M_1, M_2, \dots, M_k\}$ which are used in parallel. These parallel machines are categorized as either identical or uniform. Here, we considered uniform parallel machines for generality. This means that the processing time p_{jk} of any job j on machine k is equal to p_j/v_k . These machines are assumed to be continuously available and breakdown does not occur. Each machine can handle at most one job at a time, and each job can be processed on at most one machine at a time. Once the piston manufacturing process begins, the process cannot be stopped and resumed later. So, preemption is not allowed in this process, but idle time between two jobs is permitted.

Also, there are setup times between two consecutive parts that should be considered. S_{ij} denotes past-sequence-dependent setup time when job j follows job i . Besides, setup times are entirely separate from the processing times. On the other hand, parallel machines are mostly semi-automatic in this process, so, operator's skill affects the number of products. In this research, Biskup's position-based learning effect is employed [25]. The learning effects are just considered on the processing times. The learning effects on setup times are neglected because of short setup times. In addition, each job J_j can only be processed on a certain subset $M_j \subseteq M$ of the machines called its processing set.

The ultimate goal is to achieve just-in-time (JIT) production, The tardiness of job j is defined as $T_j = \max(0, C_j - d_j)$ and the earliness is defined as $E_j = \max(0, d_j - C_j)$, where C_j and d_j denote the completion time and the due date of job j , respectively. The objective function is to minimize the sum of earliness/tardiness costs for all jobs.

3. The proposed mathematical model

3.1 Indices

- i, j index for jobs
- r index for positions
- k index for machines

3.2 Parameters

- n number of jobs and number of positions
- m number of machines
- d_j due date of job J_j
- p_j processing time of job J_j
- V_k speed of machine k
- a learning index
- F a large positive number
- S_{ij} past-sequence-dependent setup time of job j if job i precedes job j

3.3 Decision variables

- C_j completion time of job J_j
 P_{jrk} processing time of job j when it is processed on machine k in position r
 E_j earliness of job J_j , $E_j = \max(0, d_j - C_j)$
 T_j tardiness of job J_j , $T_j = \max(0, C_j - d_j)$
 $M_{jk} \begin{cases} 1 & \text{if machine } k \text{ is eligible to process job } j \\ 0 & \text{otherwise} \end{cases}$
 $Y_{jrk} \begin{cases} 1 & \text{if job } j \text{ is scheduled on machine } k \text{ in position } r \\ 0 & \text{otherwise} \end{cases}$

3.4 The proposed mixed-integer linear programming model

Objective function:

$$\text{Min } Z = \sum_{j=1}^n (E_j + T_j)$$

Subject to:

$$P_{jrk} = \frac{p_j(r)^a}{V_k}, j = 1, \dots, n, k = 1, \dots, m, r = 1, \dots, n \quad (1)$$

$$\sum_{k=1}^m \sum_{r=1}^n Y_{jrk} = 1, j = 1, \dots, n \quad (2)$$

$$\sum_{j=1}^n Y_{jrk} \leq 1, r = 1, \dots, n, k = 1, \dots, m \quad (3)$$

$$\sum_{i \neq j, i=1}^n Y_{irk} \geq Y_{jr+1k}, j = 1, \dots, n, k = 1, \dots, m, r = 1, \dots, n-1 \quad (4)$$

$$M_{jk} \geq Y_{jrk}, j = 1, \dots, n, k = 1, \dots, m, r = 1, \dots, n \quad (5)$$

$$C_j \geq (P_{jrk} + S_{0j})Y_{jrk}, j = 1, \dots, n, k = 1, \dots, m, r = 1 \quad (6)$$

$$C_j - C_i + F(2 - Y_{irk} - Y_{jr+1k}) \geq P_{jr+1k} + S_{ij}, i = 1, \dots, n, j = 1, \dots, n, i \neq j, k = 1, m, r = 1, \dots, n-1 \quad (7)$$

$$E_j \geq d_j - C_j, j = 1, \dots, n \quad (8)$$

$$T_j \geq C_j - d_j, j = 1, \dots, n \quad (9)$$

$$E_j, T_j, C_j \geq 0, j = 1, \dots, n \quad (10)$$

$$Y_{irk} \in \{0, 1\}, i = 1, \dots, n, k = 1, m, r = 1, \dots, n \quad (11)$$

The objective function seeks to minimize the sum of E/T costs over all jobs. Constraint (1) is used to calculate the actual processing time of a job j if it is scheduled on machine k in position r . Constraint (2) ensures that each job can only take one position on one machine. Constraint (3) guarantees that each position on each machine can be assigned to at most one job. Constraint (4) ensures that there are no empty positions before a filled position. If a given job j is processed on a given machine k in position $r + 1$, a predecessor i must be processed on the same machine in position r . If job j is assigned to machine k in any position, constraint (5) guarantees machine k should be able to process job j . Constraint (6) ensures that completion time of job j is greater than or equal to the processing time of job j on machine k in the first position.

Constraint (7) establishes the relationship between the completion times of jobs i and j as long as both jobs are assigned to the same machine. Using the binary variables Y_{irk} and Y_{jr+1k} , and the large number F , this constraint enforces that there is sufficient time between the completion of jobs i and j based on the order of job precedence. Also, this restriction can consider the machine idle time if jobs i and j are assigned to machine k in positions r and $r + 1$, i.e., $Y_{irk} = 1$, $Y_{jr+1k} = 1$, the completion time of job $j(C_j)$ must be greater than or equal to the completion time of job i plus the setup time between jobs i and j and the processing time of job j in position $r + 1$. Constraints (8) and (9) calculate earliness and tardiness costs for each job, respectively. Finally, constraints (10) and (11) specify the range of decision variables.

4. Heuristic algorithms

Due to the NP-hardness of the proposed problem, developing efficient heuristic algorithms would be a good approach to achieve near-optimal solutions in a reasonable computational time. Hence, three meta-heuristics are employed in this paper; namely, genetic algorithm (GA), simulated annealing (SA), and particle swarm optimization (PSO).

The reason for using these algorithms is because of their special and strong features which made them become popular algorithms in scheduling problems. GA is a powerful and extensive applicable stochastic search and optimization technique guided by the principles of evolution and natural genetics. GA is an efficient, adaptive, and robust search process. Similar to GA, PSO is also a population-based optimizer. Although PSO does not use the crossover and mutation operators as GA does, it finds the optimum solution through individual improvement plus population cooperation and competition to form robust exploration and exploitation of the solution space in a short time. Finally, SA is an intelligent method that uses a repetitive improvement approach, but it probabilistically permits deteriorating movements to escape from local optimum solutions. This algorithm is fast and very easy to code. In addition, to find the minimum objective function for a given sequence, a heuristic algorithm named the 'JIT objective heuristic algorithm' is proposed which is used by the three proposed meta-heuristic algorithms as an objective calculator.

4.1 JIT objective heuristic algorithm

An algorithm is introduced here for the calculation of the total weighted earliness and tardiness penalties. The objective function of the proposed model (sum of earliness and tardiness penalties) is a subclass of the presented algorithm. The steps of the proposed heuristic algorithm are as follows:

1. At first, process jobs without any idle time based on the given sequence.
2. Calculate the E/T value of each job. Consider the E/T penalty of each job by the following conditions.
 - 2.1 If a processed job has earliness, regard its earliness penalty as a positive number.
 - 2.2 If a processed job has tardiness, regard its tardiness penalty as a negative number.
 - 2.3 If a job is processed at its due date, regard its tardiness penalty as a negative number.

Therefore, until now, the procedures of steps 1, 2, and 3 are shown in the following table:

Table 1. The primitive calculations and positions of values

Sequence	J_2	J_5	J_1	J_3	J_4
Status	γ_2	γ_5	γ_1	γ_3	γ_4
Penalty	β_2	β_6	β_1	β_3	β_4

Where the ‘sequence’ row shows a given sequence of jobs (here, this sequence is hypothetical); the ‘status’ row indicates the E/T value of jobs, and the ‘penalty’ row demonstrates the E/T penalty of jobs.

In step 1, calculate the completion time of each job for the given sequence without considering any idle time between jobs using the following equation:

$$C_i = \sum_k P_{[k]} \forall i, k \quad (12)$$

In step 2, calculate the E/T value for each job using the following equation:

$$\gamma_i = d_i - \sum_k P_{[k]} \forall i, k \quad (13)$$

Where $p_{[k]}$ denotes the normal processing time of the job scheduled in the k th position in the sequence and d_i denotes the due date of job i . For example, $\gamma_5 = d_5 - (p_2 + p_5)$, $\gamma_1 = d_1 - (p_2 + p_5 + p_1)$, and $\gamma_3 = d_3 - (p_2 + p_5 + p_1 + p_3)$.

3. Validation test: In the status row of the remained jobs, select the first job with earliness value and add its penalty to the penalty of the next jobs. This procedure continues until the summation reaches the first zero value or the first negative value. If at any stage of the summation operation, the sum of penalties remains positive until the last job, this set is a valid set, then go to step 3.2; otherwise, it is called ‘invalid set’, then go to step 3.1. Therefore, an invalid set starts with a first selected job having earliness value and ends with a job that sum of the E/T penalties of jobs until then reaches the first zero value or the first negative value.

3.1 Exclude the invalid set from algorithm calculation. If any job with earliness value exists after the invalid set, then go to step 3; otherwise, go to step 4.

3.2 In the condition row of the valid set, select the minimum earliness value and then subtract this value from all the values of the valid set. Then, update the penalty row for each job by rules of 2.1, 2.2, and 2.3. If another earliness value still exists in the remained jobs, then go to step 3; otherwise, go to step 4.

4. Calculate the completion time of each job and the total cost using the following equations.

$$C_i = d_i - \gamma_i \quad (14)$$

$$\text{Total cost} = \sum_{i=1}^n \gamma_i \times \beta_i \quad (15)$$

Where n , C_i , and γ_i refer to the number of jobs, the completion time of each job, and E/T of each job, respectively.

5. End of the algorithm

The proposed heuristic algorithm is depicted in Figure 1.

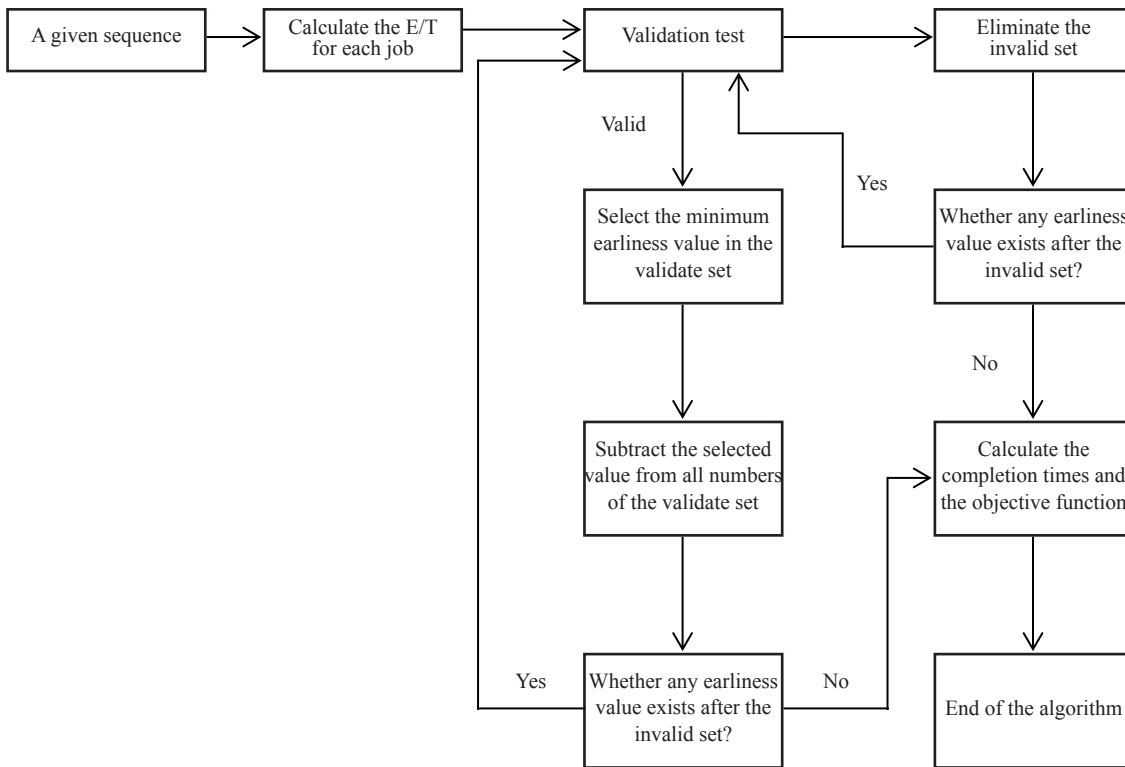


Figure 1. The proposed JIT objective heuristic algorithm

For clarification, an instance is investigated. Assume that a given sequence of 6 jobs is selected by a selection strategy which a meta-heuristic may apply in iterations. The selected sequence of jobs is equal to $J_3, J_6, J_2, J_4, J_1, J_5$. Also, the values of the parameters are shown in Table 2. Furthermore, Figure 2 shows the scheduling of the given sequence based on the proposed heuristic algorithm.

Table 2. Problem parameters

job_i	p_i	d_i	a_i	β_i
J_1	6	29	2	1
J_2	7	11	2	3
J_3	2	4	4	3
J_4	3	23	4	2
J_5	4	30	1	1
J_6	8	9	3	2

Step 1.

Based on the given sequence, jobs are processed without any idle time.

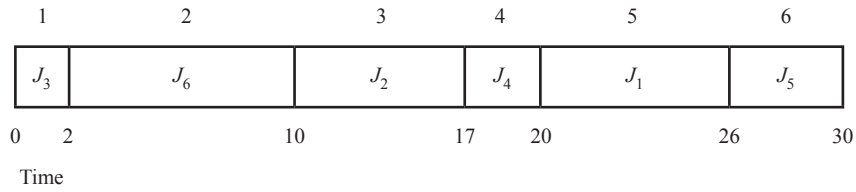


Figure 2. Jobs scheduled without any idle time

Step 2.

Based on the jobs processed at the previous step, the E/T is calculated for each job and the results are listed in Table 3.

Table 3. Results of jobs scheduled in step 1

Jobs	J_3	J_6	J_2	J_4	J_1	J_5
Condition	2	-1	-6	2	3	0
Penalty	4	-2	-3	4	2	-1

In Table 3, it can be seen that for example, J_4 has 2 units of earliness with the weight of 4; or for J_2 , the number (-6) shows that J_2 has 6 units of tardiness and (-3) is the weight of its tardiness that is denoted as a negative value as mentioned in step 2.2. Also, the value of the condition row of J_5 is equal to zero which means this job is processed on its due date, and the tardiness weight for it is considered as mentioned in step 2.3.

Step 3.

The first job with earliness is J_3 . So, the sum of the penalties of J_3 to J_5 is calculated. As shown in Table 4, it can be observed that the sum of the penalties of J_3 , J_6 , and J_2 are equal to $4 - 2 - 3 = -1$ which means this set is invalid. Thus, go to step 3.1.

Step 3.1.

The invalid set is excluded from the algorithm because this set doesn't assure to improve the objective function. Thus, as displayed in Table 5, the remaining jobs are J_4 , J_1 , and J_5 . Also, there is at least one job with earliness. So, go to step 3 again.

Table 4. The first invalid set

Jobs	J_3	J_6	J_2
Condition	2	-1	-6
Penalty	4	-2	-3

Table 5. Remained jobs

Jobs	J_4	J_1	J_5
Condition	2	3	0
Penalty	4	2	-1

Step 3.

J_4 is the first job with earliness. So, the sum of the penalties of J_4 to J_5 is calculated. The sum of the penalties of J_4 , J_1 , and J_5 is equal to $4 + 2 - 1 = 5$ which means this set is valid (see Table 6). Thus, go to step 3.2.

Table 6. The first valid set

Jobs	J_4	J_1	J_5
Condition	2	3	0
Penalty	4	2	-1

Step 3.2.

In the valid set as it can be seen in Table 7, the minimum positive value is 2 that belongs to J_4 . So, this value is subtracted from all of the condition values of the valid set. With regards to the changes of the condition value for each job, the earliness or tardiness penalty is returned. The Table 7 is updated as follows:

Table 7. The updated first valid set

Jobs	J_4	J_1	J_5
Condition	0	1	-2
Penalty	-2	2	-1

Since at least a positive value still exists, return to step 3.

Step 3.

Here, the first and only positive value is 1 that belongs to J_1 . So, the validation of the set of J_1 to J_5 is examined. The sum of the penalties of J_1 and J_5 is equal to $2 - 1 = 1$ which means this set is valid (see Table 8). Thus, go to step 3.2. Furthermore, the value of the condition row of J_4 is equal to zero. So, this job is also excluded from the algorithm (see Table 9). Because this job doesn't improve the objective value anymore.

Table 8. The second valid set

Jobs	J_1	J_5
Condition	1	-2
Penalty	2	-1

Table 9. The excluded job

Jobs	J_4
Condition	0
Penalty	-2

Step 3.2

So, this value is subtracted from all of the condition values of the valid set. Thus, the updated valid set is shown in Table 10.

Table 10. The second updated valid set

Jobs	J_1	J_5
Condition	0	-3
Penalty	-1	-1

It can be seen, there is no more positive value. So, we go to step 4. In this step, the completion time of each job and the total cost are calculated. Also, the final values are shown in Table 11. The final table is composed of Tables 4, 9, and 10.

Table 11. The final results

Jobs	J_3	J_6	J_2	J_4	J_1	J_5
Condition	2	-1	-6	0	0	-3
Penalty	4	-2	-3	-2	-1	-1

Step 4.

According to formulas (14) and (15), the completion time of each job and total cost are calculated.

$$C_3 = 4 - 2 = 2, C_6 = 9 - (-1) = 10, C_2 = 11 - (-6) = 17, C_4 = 23 - 0 = 23, C_1 = 29 - 0 = 29 \text{ and } C_5 = 30 - (-3) = 33$$

$$\text{Total coast} = 2 \times 4 + (-1) \times (-2) + (-6) \times (-3) + 0 \times (-2) + 0 \times (-1) + (-3) \times (-1) = 31$$

Step 5.

End of the algorithm

In section 5, 24 small size test problems are defined in Table 12 to validate the presented JIT objective heuristic algorithm in obtaining optimum solutions.

4.2 Genetic algorithm

GAs are intelligent random search strategies with the ability to find near-optimal solutions in complex search spaces without derivative information. The basic concepts of GA have been described by the investigation carried out by Holland [26]. The components of the GA applied to solve the proposed problem are described as follows.

4.2.1 Chromosome representation and the initial population

Chromosome representation has a crucial impact on GA's performance. A good representation scheme is necessary to describe the specific characteristics of a given problem in detail. Here, each chromosome is a matrix with (m, n) dimensions which m and n refer to the number of machines and the number of jobs, respectively. So, each chromosome is divided into $n \times m$ positions. An example is presented to illustrate the procedure. Four jobs on 2 machines are considered. The processing set of J_1 is $\{M_1\}$, for J_2 is $\{M_2\}$ and the processing set of J_3 and J_4 are $\{M_1, M_2\}$. Figure

3 shows several feasible chromosomes. The empty positions are shown by a value of zero. Furthermore, according to constraints (4) and (5) in the proposed mathematical model, chromosome representation in Figures 4 and 5 is not correct. The GA is formed by a population of P_{size} individuals, where each individual consists of m rows (machines) and n columns (jobs).

1	0	0	0
3	2	4	0

1	3	0	0
2	4	0	0

1	3	4	0
2	0	0	0

Figure 3. Feasible chromosomes

0	0	1	0
3	0	4	2

Figure 4. Infeasible chromosome

1	2	0	0
3	4	0	0

Figure 5. Infeasible chromosome

In a genetic algorithm, it is also common to randomly generate the initial population. The same way is employed in this study to generate individual chromosomes.

4.2.2 Fitness function

The objective here is to minimize the sum of E/T. By calculating the completion time, earliness and tardiness of each job are obtained applying equations (16) and (17), respectively.

$$E_i = \max(0, d_i - c_i) \quad (16)$$

$$T_i = \max(0, c_i - d_i) \quad (17)$$

Thus, the fitness function of each chromosome can be calculated as follows:

$$F = \sum_{i=1}^n (E_i + T_i) \quad (18)$$

4.2.3 Selection strategies

In selection strategies, it is important to prevent the algorithm from converging quickly to a local minimum. The roulette-wheel method uses a probability distribution for selection in which the selection probability of a given string is proportional to its fitness. Thus, a more fitted string is more likely to be selected, but a bad string still has its chance. In this paper, the roulette-wheel method is applied to select the parents for crossover and mutation. The probability p_i of selecting a particular individual i in the minimization problem is given by:

$$p_i = \frac{\frac{1}{f_i}}{\sum_{j=1}^N \frac{1}{f_j}} \quad (19)$$

Where f_i is the fitness of individual i and N is the size of the population.

4.2.4 Crossover

The crossover operator is an important component of GA. A crossover operation is employed to generate new offspring from randomly selected pairs of parents by uniting them. One of the most used crossover operators is the One Point Order Crossover adapted to the parallel machine case; such that, for each machine, one-point p is randomly selected from parent 1, and jobs from the 1st position to the p th position are copied to the offspring. In Figure 6, an example of 8 jobs and two machines is given. Two parents are shown and for each machine, a point p is selected. Point p_1 (machine 1) is set to 2 and point p_2 (machine 2) is set to 3. At first, the offspring is formed with the genes (jobs) of parent A from position 1 to 2 on machine 1, and from position 1 to 3 on machine 2. Then, the genes of parent B which do not exist in the offspring, are inserted in the next positions of the offspring on the same machine. If a position with the value of zero is selected for point p , then another selection must be done until a non-zero position is found.

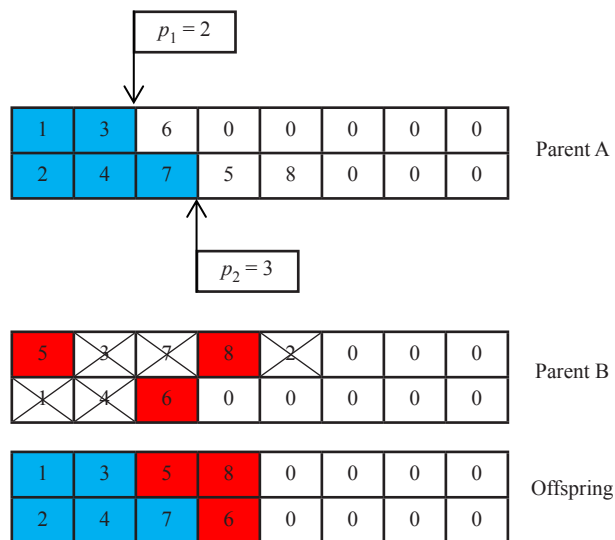


Figure 6. Crossover operator

4.2.5 Mutation and reproduction

The purpose of the mutation is to ensure that diversity is maintained in the population. It creates a new chromosome by altering the place of the genes. The mutation operator is applied individually to each chromosome. In

this paper, the swapping method is used. In this method, two random genes are selected and their positions are swapped. The procedure is as follows. Searching space for selecting the first gene is the interval $[1, n]$ on each machine that n refers to the number of jobs scheduled on that machine, and the second gene is selected from $[1, n + 1]$. These rules improve the performance of the mutation operator by searching the effective space. Also, the searching spaces of the first gene and the second gene are shown in Figure 7 and Figure 8, respectively. Therefore, based on the defined rule, one random gene is selected and is kept as a first gene, then another gene is selected randomly. If two genes are capable of being exchanged under processing set restrictions, then these two genes are swapped; otherwise, another second gene is selected randomly. This procedure will continue until the first selected gene can swap with another gene. Figure 9 shows this procedure. In this method, if the second gene is a filled position by a job, then the mutated chromosome remains feasible. But, if the second gene is in an empty position, then the mutated chromosome is infeasible. So, in the second case, the procedure is as follows. The first gene is inserted in the second gene's position. The jobs after the first gene will be shifted one position to the left at the same machine and the second gene will be inserted after the scheduled jobs on that machine. This procedure is depicted by two examples in Figure 10.

6	3	5
2	4	7
8	1	

Figure 7. Searching space for the first gene

6	3	5	0
2	4	7	0
8	1	0	

Figure 8. Searching space for the second gene



Figure 9. The first case of the mutation operator

Reproduction plays an important role in the successful convergence of the algorithm. It is performed by the elitism procedure. In this strategy, the best individuals which have minimum fitness are kept and passed on to the next generation. Here, the number of best individuals kept is a parameter of the algorithm.

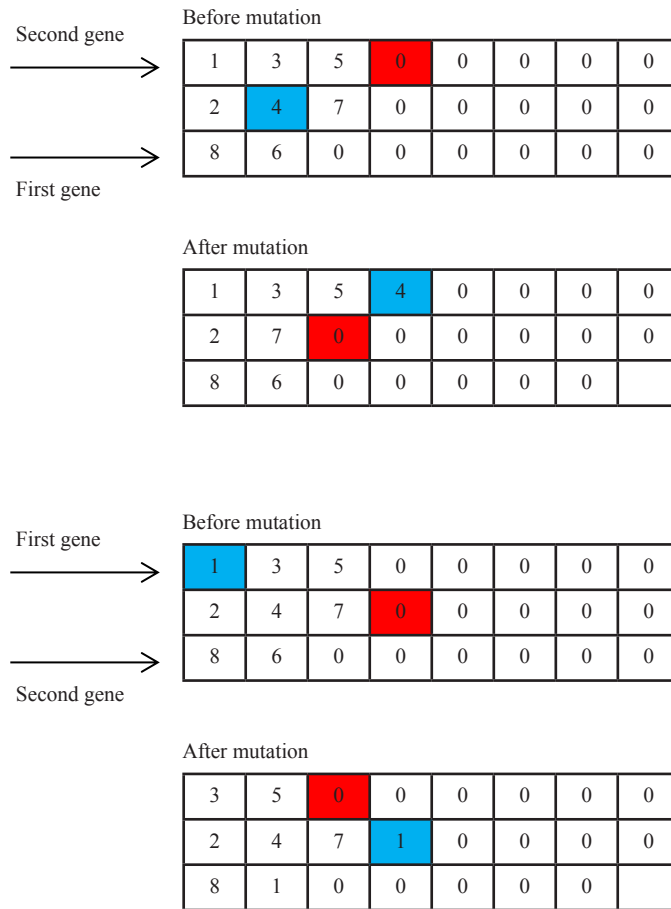


Figure 10. The second case of the mutation operator

4.3 Simulated annealing (SA) algorithm

Simulated annealing (SA) provides a mechanism to escape from local optima by allowing hill-climbing moves. Its ease of implementation and convergence properties have made it a popular approach over the past two decades [27]. A standard SA begins with an initial random solution and an appropriate high temperature (T_0). This temperature is periodically reduced by some temperature functions until the temperature becomes near to zero as the method progress (T_f). The main components of SA for implementation are as follows.

4.3.1 Creating the initial answer and neighborhood search

Based on the given explanation in section 4.2.1, a random answer is created. To generate a neighborhood of the current solution, one machine is randomly selected, then the scheduled job with the worsening E/T is selected at the same machine and then swapped with a random place under defined considerations in section 4.2.5. The randomly generated neighbor solution becomes a new solution if it improves the objective function; else, the neighborhood solution becomes a new solution with an appropriate probability based on $p = \exp\left(\frac{-\Delta E}{T}\right)$, where ΔE is a measure to which neighbor solution becomes worse than the current solution, and T is the temperature parameter of the current iteration. A random number between 0 and 1 is generated through a uniform distribution. If $p > \text{rand}(0, 1)$, then the answer has deteriorated. Otherwise, another neighborhood will be chosen.

4.3.2 Initial and final temperature

Temperature as a parameter plays an important role in rejecting or accepting the objective function. It should be high enough to make an equal chance for all points of the search space and simultaneously, it should not be very too high to avoid carrying out a lot of unnecessary searches in high temperatures. In this paper, the initial temperature is determined experimentally and the final temperature is obtained by $T_f = \beta T_0$.

4.3.3 Cooling ratio

The fundamental purpose of employing the annealing schedule is to control the behavior of SA. There are distinctive cooling ratios used in the SA literature. In this paper, the annealing schedule is calculated as follows:

$$A = \frac{(T_0 - T_N)(N + 1)}{N} \quad (20)$$

$$B = (T_0 - A) \quad (21)$$

$$T_i = \frac{A}{i + 1} + B \quad (22)$$

Where N is the number of iterations to meet the final temperature, T_N is the temperature in the final iteration that is equal to T_f and T_i is the temperature in iteration i .

4.4 Particle swarm optimization (PSO)

Introduced by Kennedy and Eberhart [28], PSO is a stochastic global optimization technique inspired by the social behavior of bird flocking. In PSO, every single solution is a particle in the search space and the set of particles forms the swarm. The position of i th particle in d -dimensional search space is represented by a d -dimensional vector $X_i = (x_{i1}, x_{i2}, \dots, x_{id})(i = 1, 2, \dots, m)$. Also, the i th particle's velocity is a d -dimensional vector, denoted by $V_i = (v_{i1}, v_{i2}, \dots, v_{id})(i = 1, 2, \dots, m)$. The best position of the particle i obtained up to iteration t is denoted by $P_i = (p_{i1}(t), p_{i2}(t), \dots, p_{id}(t))$, and the best position of the swarm in iteration t is denoted by $P_g = (p_{g1}(t), p_{g2}(t), \dots, p_{gd}(t))$, respectively. In any iteration, positions and velocities are updated by the equations (23) and (24):

$$V_{id}(t + 1) = W V_{id}(t) + C_1 R_1 (P_{id}(t) - X_{id}(t)) + C_2 R_2 (P_{gd}(t) - X_{id}(t)) \quad (23)$$

$$X_{id}(t + 1) = X_{id}(t) + V_{id}(t + 1) \quad (24)$$

Where W is the inertia weight that controls the influence of the previous velocity of the particle. C_1 and C_2 are acceleration constants that drive particles towards local and global best positions. R_1 and R_2 are two random numbers within the range of $[0, 1]$. Eq. (23) is used to calculate the new velocities for particles and Eq. (24) updates each particle's position in the search space. This process is repeated until a user-defined stopping criterion is reached. In this paper, a linear function is used to set the inertia weight. This function is as follows:

$$W = W_{\max} - \frac{W_{\max} - W_{\min}}{t_{\max}} t \quad (25)$$

Where W_{\max} and W_{\min} are set to 1 and 0, respectively. In addition, t shows the current iteration number and t_{\max} shows the maximum iteration number.

Generally, to control the excessive roaming of particles outside the search space, the velocity of each particle

can be restricted to the interval $[V_{min}V_{max}]$. Within the scope of this paper, the V_{min} and the V_{max} are equal to 0 and 1, respectively. Also, random particles are generated based on the given explanation in section 4.2.1. The overall structure of the proposed algorithm is shown in Algorithm 1.

Algorithm 1. The process of particle swarm optimization
 Initialize a population of particles with random positions and velocities
 Begin
 repeat
 for each particle i do
 Update the position and the velocity of particle i
 Evaluate the fitness value of particle i
 If the current value of particle i is better than the p_{id}
 Then set p_{id} to the current value
 If p_{id} is better than the global best position
 Then set p_{gd} to the current particle value
 end for
 until a termination criterion is met

5. Experimental results

Table 12. Name of generated test problems

n^a	m^b	$t = 0.2$			$t = 0.4$		
		$R = 0.6$	$R = 0.8$	$R = 1$	$R = 0.6$	$R = 0.8$	$R = 1$
6	2	Num01	Num02	Num03	Num04	Num05	Num06
6	4	Num07	Num08	Num09	Num10	Num11	Num12
8	2	Num13	Num14	Num15	Num16	Num17	Num18
8	4	Num19	Num20	Num21	Num22	Num23	Num24
14	4	Num25	Num26	Num27	Num28	Num29	Num30
14	5	Num31	Num32	Num33	Num34	Num35	Num36
16	4	Num37	Num38	Num39	Num40	Num41	Num42
16	5	Num43	Num44	Num45	Num46	Num47	Num48
30	5	Num49	Num50	Num51	Num52	Num53	Num54
30	6	Num55	Num56	Num57	Num58	Num59	Num60
40	5	Num61	Num62	Num63	Num64	Num65	Num66
40	6	Num67	Num68	Num69	Num70	Num71	Num72

n^a = number of jobs, m^b = number of machines

The objective of the computational experiments described in this section is to evaluate the performance of the proposed algorithms. Therefore, a set of test problems is needed for comparing the results of the proposed meta-heuristic algorithms. To present the efficiency of the proposed approaches, problems with different sizes are considered. Small, medium and large size problems consist of 6, 8 jobs on 2, 4 machines, 14, 16 jobs on 4, 5 machines, and 30, 40 jobs on 5, 6 machines, respectively. Each job $j_i (i = 1, 2, \dots, n)$ has a randomly set of machines $b_i = U(1, m)$ to which it can be assigned. Speeds of machines are selected from the set $\{1, 0.6, 0.8\}$. Processing times are generated from the

discrete uniform distribution [1, 25] and setup times are uniformly distributed, ranging from 20% to 40% of the mean of the processing times. Setup time matrices are asymmetric. It is seen that in manufacturing systems and especially in assembly lines, the learning rate fits the 80% learning curve. In this paper, the same rate is also used as a learning rate. Due dates of jobs are generated from a uniform distribution $[C_{max}(1-t-\frac{R}{2}), C_{max}(1-t+\frac{R}{2})]$ as suggested by Potts and Wassenhove [29], where $C_{max} = \frac{P+ns}{m}$, $P = \sum_{i=1}^n p_i$, n and m refer to the number of jobs and machines, respectively. s is the average setup time ($\frac{s_{min} + s_{max}}{2}$), s_{min} and s_{max} are equal to the lower and upper bound of the setup times interval, respectively. This method is controlled by two parameters, t and R . t is the priority factor that takes the values $\{0.2, 0.4\}$, and R is the due date range factor that takes the values $\{0.6, 0.8, 1\}$. So, there are six instances considered for each problem size. Each problem runs 10 times. Thus, for each meta-heuristic, there are 720 runs in total. The name of generated test problems with variable parameters is given in Table 12. The mathematical formulation (MILP) is solved using the global solver of LINGO 9.0. The proposed meta-heuristic algorithms are coded and run using MATLAB 7.11. All experimental tests are carried out on a computer Intel (R) core i5 (2.67 GHz CPU) with 512 MB RAM.

Table 13. The computational results of the proposed algorithms for small-sized problems

Num	LINGO			GA			SA			PSO		
	OPT	B	M	W	B	M	W	B	M	W		
01	19.08	19.08	19.08	19.08	19.08	19.08	19.08	19.08	19.08	19.08	19.08	
02	10.44	10.44	10.44	10.44	10.44	10.44	10.44	10.44	10.44	10.44	10.44	
03	16.72	16.72	16.72	16.72	16.72	16.72	16.72	16.72	16.72	16.72	16.72	
04	33.33	33.33	33.33	33.33	33.33	33.33	33.33	33.33	33.33	33.33	33.33	
05	30.84	30.84	30.84	30.84	30.84	30.84	30.84	30.84	30.84	30.84	30.84	
06	16.63	16.63	16.63	16.63	16.63	16.63	16.63	16.63	16.63	16.63	16.63	
07	6.36	6.36	7.56	9.36	6.36	8.62	12.03	6.36	6.96	9.36	9.36	
08	9.76	9.76	11.82	12.91	9.76	13.06	13.96	9.76	11.50	13.96	13.96	
09	22.76	22.76	23.09	26.11	22.76	23.32	26.11	22.76	23.32	26.11	26.11	
10	0.00	0.00	0.37	0.83	0.00	0.28	0.56	0.00	0.34	0.56	0.56	
11	4.56	4.56	7.38	16.31	4.56	6.16	16.55	4.56	5.88	16.55	16.55	
12	15.96	15.96	18.01	22.72	15.96	15.96	15.96	15.96	16.66	19.48	19.48	
13	46.16	46.16	47.50	49.75	46.16	49.85	56.17	46.16	47.11	49.75	49.75	
14	54.91	54.91	54.91	54.91	54.91	55.56	57.48	54.91	54.91	54.91	54.91	
15	34.16	34.16	39.01	51.20	34.16	39.32	46.13	34.16	38.39	46.13	46.13	
16	32.92	32.92	35.19	38.96	32.92	37.45	40.23	32.92	34.76	38.96	38.96	
17	23.28	23.28	26.30	26.73	23.28	23.71	26.73	23.28	25.87	26.73	26.73	
18	74.62	74.62	74.62	74.62	74.62	82.51	94.62	74.62	74.62	74.62	74.62	
19	8.40	8.40	11.60	19.55	8.40	12.93	21.12	8.40	11.40	19.55	19.55	
20	10.52	10.52	12.51	22.43	10.52	13.82	23.92	10.52	12.64	22.05	22.05	
21	0.99	0.99	1.95	4.72	0.99	2.45	12.48	0.99	1.83	7.70	7.70	
22	13.24	13.24	14.79	18.20	13.24	14.23	16.70	13.24	14.01	16.25	16.25	
23	8.99	8.99	11.16	19.18	8.99	10.21	17.07	8.99	10.01	17.49	17.49	
24	10.44	10.44	11.36	16.94	10.44	11.61	18.31	10.44	11.23	15.36	15.36	
mean	21.04	21.04	22.34	25.52	21.04	22.84	26.80	21.04	22.02	25.10	25.10	

The results of the small, medium and large size problems solved by the proposed methods are presented in Tables 13, 14, and 15, respectively. For each test case, the best, mean, and worst solutions of the proposed meta-heuristics are given, where B denotes the best solution, M represents the mean solutions and W indicates the worst solutions.

Table 14. The computational results of the proposed algorithms for medium-sized problems

Num	LINGO		GA			SA			PSO		
	OPT	B	M	W	B	M	W	B	M	W	
25	-	25.04	30.17	43.22	25.04	29.66	34.53	25.04	27.26	31.35	
26	-	25.24	30.16	46.46	26.16	29.24	40.48	25.24	28.92	44.07	
27	-	12.33	14.22	21.90	16.06	18.45	26.64	12.33	13.85	20.75	
28	-	31.60	35.59	41.08	35.23	38.33	40.79	31.6	34.26	39.58	
29	-	18.12	20.18	21.40	15.01	16.77	25.71	13.52	14.88	21.40	
30	-	25.48	30.22	48.92	24.73	28.01	40.29	25.48	27.52	39.24	
31	-	46.02	50.62	60.98	45.97	50.11	56.42	44.16	47.69	55.79	
32	-	33.47	36.14	38.56	27.79	31.40	38.33	22.63	24.44	34.26	
33	-	10.32	11.62	15.66	11.99	13.64	17.32	12.38	12.93	14.13	
34	-	34.23	45.60	57.49	29.89	34.07	51.39	31.05	35.71	48.86	
35	-	18.54	20.02	25.97	18.26	21.36	31.88	17.25	18.63	23.92	
36	-	32.02	34.58	37.76	32.02	37.78	42.61	32.02	34.90	38.55	
37	-	64.90	70.74	74.01	69.20	78.19	82.21	51.19	54.77	62.21	
38	-	35.47	39.01	42.22	38.27	43.63	50.89	35.00	37.80	42.22	
39	-	24.40	26.59	33.24	24.70	27.66	38.42	21.76	23.72	31.25	
40	-	44.12	46.32	48.25	42.00	46.62	51.33	38.94	42.44	47.36	
41	-	42.81	46.23	58.31	40.07	44.18	50.07	41.26	45.79	55.74	
42	-	48.70	52.10	59.62	50.94	56.04	63.25	51.28	55.89	60.03	
43	-	63.57	69.29	75.53	66.49	74.47	80.79	63.57	67.38	72.19	
44	-	53.78	55.39	56.53	56.55	61.90	64.09	53.78	55.93	57.10	
45	-	30.59	34.57	40.48	30.59	36.71	50.23	30.59	33.65	38.06	
46	-	53.75	57.51	73.47	47.29	53.43	66.94	47.29	52.42	63.61	
47	-	45.43	49.06	68.72	45.74	50.77	65.29	45.43	49.97	63.29	
48	-	20.00	22.06	31.49	20.00	24.01	36.41	20.00	22.20	28.44	
mean	-	35.00	38.67	46.72	35.00	39.43	47.76	33.03	35.96	43.06	

Table 15. The computational results of the proposed algorithms for large-sized problems

Num	LINGO	GA			SA			PSO		
	OPT	B	M	W	B	M	W	B	M	W
49	-	112.41	117.86	136.62	112.41	120.26	140.87	105.17	111.48	136.62
50	-	45.13	48.67	60.58	45.32	50.21	64.23	41.13	44.01	58.18
51	-	24.49	27.33	30.24	21.76	23.09	27.73	22.45	24.16	28.23
52	-	86.96	93.92	110.11	100.34	106.82	115.12	83.31	89.14	100.27
53	-	88.79	94.12	104.43	90.33	97.56	110.35	88.33	94.51	107.28
54	-	29.18	31.77	35.90	25.20	26.73	30.62	27.29	28.71	31.37
55	-	124.33	134.28	150.22	124.63	133.35	145.16	124.06	130.26	143.82
56	-	43.59	48.61	57.24	40.90	45.96	51.53	33.62	36.65	44.49
57	-	26.94	29.09	33.58	28.44	32.44	38.55	28.13	29.85	33.58
58	-	171.11	174.95	180.68	161.92	170.49	186.96	150.52	158.07	173.74
59	-	52.25	57.03	68.21	51.35	56.16	68.21	45.15	48.76	60.15
60	-	19.18	22.01	26.67	21.51	23.44	26.20	17.77	19.19	22.40
61	-	260.69	271.12	283.25	282.60	291.09	301.44	239.60	246.79	256.82
62	-	107.85	114.48	125.87	115.62	122.20	140.27	102.21	108.39	125.87
63	-	76.20	82.434	90.63	69.58	73.24	86.24	73.66	77.11	88.26
64	-	217.41	243.50	253.22	209.67	226.44	244.91	191.41	202.89	236.92
65	-	53.55	56.44	64.63	51.68	54.93	60.62	53.86	56.61	65.02
66	-	84.36	92.80	98.54	81.63	87.34	94.22	79.62	85.43	100.87
67	-	198.02	211.36	230.24	193.62	203.24	218.10	200.12	207.37	225.18
68	-	87.57	98.08	115.37	87.57	95.45	120.47	80.81	86.47	115.37
69	-	51.86	55.49	68.23	53.18	57.08	72.26	53.44	56.87	68.23
70	-	163.66	180.03	198.95	139.64	155.00	175.44	130.47	136.83	151.88
71	-	78.13	86.72	87.91	63.32	68.39	75.14	63.83	68.04	75.14
72	-	26.04	28.38	33.42	27.57	32.19	36.73	25.01	25.98	27.11
mean	-	92.90	100.02	110.19	91.65	98.04	109.64	85.87	90.56	103.2

5.1 Parameter setting

In this paper, the Taguchi method is utilized to determine the values of the parameters of GA, PSO, and SA. After several pretests, each of these parameters is set at three levels. Test problems are made of a variable number of jobs and machines. Each experiment is implemented five times. After completing the tests, Taguchi analysis is applied for

different values of parameters. The best values of the parameters of GA, PSO, and SA are listed in Table 16.

Table 16. Parameter settings of GA, PSO, and SA

GA	PSO	SA
The iteration number is 85	The iteration number is 80	The maximum iteration number is 100
The population size is 80	The particle number is 50	The final temperature coefficient of determination (β) is 0.01
The crossover rate is 0.6	The cognition learning factor: $c_1 = 2, c_2 = 2$	The iteration number in each temperature is 10
The mutation rate is 0.15	-	The number of neighbors to determine the initial temperature is 90

5.2 Performance evaluation

According to Tables 13, 14, and 15, medium and large size problems cannot be solved using LINGO in a reasonable computational time. In order to find optimum or near-optimal solutions for medium and large-size problems in a reasonable time, heuristic algorithms are employed and two metrics are applied to evaluate the performance of the proposed algorithms. Since small size problems can be solved optimally by LINGO, the percentage relative error (PRE) is employed to assess the performance of the algorithms for small size problems. PRE is as follows:

$$PRE = \frac{\text{Heuristic solution} - \text{Optimal solution}}{\text{Optimal solution}} \times 100 \quad (26)$$

For medium and large size problems, the relative percentage deviation (RPD) is employed to compare the efficiency of three proposed meta-heuristics. The RPD is computed as follows:

$$RPD = \frac{\text{Heuristic solution} - \text{Best heuristic solution}}{\text{Best heuristic solution}} \times 100 \quad (27)$$

The average errors of these meta-heuristic methods and their computational times for the small, medium, and large size problems are given in Tables 17, 18, and 19, respectively. In addition, the average computational time, the average PRE_{avg} , and RPD_{avg} for each job group are summarized in Table 20. Furthermore, after computing RPD, in order to make better comparisons between the proposed meta-heuristics, the analysis of variance (ANOVA) is conducted and relative percentage deviation means plot with least significant difference (LSD) intervals at a 95% confidence level are tested. Results are shown in Figure 11. The LSD test consists of two phases. In the first phase, it uses the F statistic for testing overall equality ($H_0: \mu_i = \mu_j$, for all $i \neq j$). If the null hypothesis (H_0) is not rejected ($p - value > 0.05$), then all means are expressed to be equal and the LSD test is finished. If the null hypothesis (H_0) is rejected ($p - value \leq 0.05$), then the experimenter proceeds to phase 2, where all pairs of means (μ_i and μ_j) are separately tested for equality using α -level t -tests. If any of these t -tests rejects the null hypothesis, then the two corresponding means are expressed to be unequal.

Table 13 shows that the three proposed meta-heuristics are efficient and able to reach optimum solutions in small-sized instances. Hence, results prove that the proposed JIT objective heuristic algorithm which is applied as an objective calculator in GA, SA, and PSO is valid because in the first 24 test problems (small size problems), all the three proposed meta-heuristics obtain the same optimum solutions calculated by LINGO. According to Table 17, the average PRE_{avg} of PSO is 12.24% that is less than the other two algorithms. Also, the computational times of the proposed meta-heuristic algorithms are so short that they can be neglected. Furthermore, according to the summarized results in Table 20, the

comparison of the average RPD_{avg} of job groups between GA, SA, and PSO is illustrated in Figure 12.

Table 17. Average PRE errors and computational times of the proposed meta-heuristics for the small cases

Num	GA		SA		PSO	
	PRE_{avg}	CPU_{avg}^a	PRE_{avg}	CPU_{avg}^a	PRE_{avg}	CPU_{avg}^a
01	0.00	0.28	0.00	0.14	0.00	0.27
02	0.00	0.27	0.00	0.14	0.00	0.25
03	0.00	0.26	0.00	0.14	0.00	0.25
04	0.00	0.65	0.00	0.59	0.00	0.24
05	0.00	0.33	0.00	0.14	0.00	0.29
06	0.00	0.30	0.00	0.14	0.00	0.28
07	18.87	1.30	35.53	0.55	9.43	1.00
08	21.11	0.75	33.81	0.31	17.83	0.55
09	1.45	0.30	2.46	0.19	2.46	0.27
10	37.00	0.32	28.00	0.16	34.00	0.27
11	61.84	1.30	35.09	0.68	28.95	0.74
12	12.84	0.30	0.00	0.16	4.39	0.26
13	2.90	1.30	7.99	0.58	2.03	1.00
14	0.00	1.10	1.18	0.24	0.00	0.68
15	14.20	0.43	15.10	0.31	12.38	0.41
16	6.90	0.58	13.76	0.52	5.59	0.44
17	12.97	1.37	1.84	0.26	11.12	1.35
18	0.00	1.67	10.57	1.23	0.00	1.43
19	38.09	1.56	53.92	1.34	35.71	1.40
20	18.92	1.28	31.37	1.08	20.15	1.17
21	96.97	1.55	147.47	1.00	84.85	1.20
22	11.70	1.31	7.48	1.08	5.81	1.12
23	24.14	1.63	13.57	0.63	11.35	1.72
24	8.81	1.70	11.21	0.90	7.57	1.52
mean	16.20	0.91	18.77	0.52	12.24	0.75

^aComputational time (second).

Table 18. Average RPD errors and computational times of the proposed meta-heuristics for the medium cases

Num	GA		SA		PSO	
	RPD _{avg}	CPU _{avg} ^a	RPD _{avg}	CPU _{avg} ^a	RPD _{avg}	CPU _{avg} ^a
25	20.49	9.97	18.45	5.01	8.87	9.32
26	19.49	8.80	15.85	5.11	14.58	8.55
27	15.33	8.84	49.64	5.28	12.33	8.31
28	12.63	8.51	21.30	4.92	8.42	8.01
29	49.26	8.50	24.04	4.98	10.06	7.83
30	22.20	8.57	13.26	5.75	11.28	7.84
31	14.63	8.59	13.47	5.47	7.99	7.16
32	59.67	8.43	38.75	4.99	8.00	7.88
33	12.60	7.74	32.17	5.32	25.29	7.05
34	52.56	7.07	13.98	5.21	19.47	6.26
35	16.07	6.84	23.83	5.01	8.00	6.29
36	7.99	7.45	17.99	4.81	8.99	6.42
37	38.19	11.53	52.74	5.20	6.99	10.26
38	11.48	12.01	24.66	5.10	8.00	11.10
39	22.20	12.00	27.11	5.22	9.01	8.00
40	18.95	13.25	19.72	5.91	8.99	10.01
41	15.37	12.50	10.26	5.44	14.27	9.66
42	6.98	12.19	15.07	5.32	14.76	10.75
43	8.99	9.86	17.15	5.71	5.99	8.68
44	2.99	11.00	15.10	5.98	3.99	9.11
45	13.01	10.55	20.01	5.94	10.00	9.38
46	21.61	10.30	12.98	5.81	10.84	9.57
47	7.75	9.55	11.51	5.89	9.75	9.02
48	10.30	10.76	20.05	5.60	11.00	8.66
mean	20.03	9.78	22.05	5.37	10.70	8.55

^aComputational time (second).

Table 19. Average *RPD* errors and computational times of the proposed meta-heuristics for the large cases

Num	GA		SA		PSO	
	RPD _{avg}	CPU _{avg} ^a	RPD _{avg}	CPU _{avg} ^a	RPD _{avg}	CPU _{avg} ^a
49	12.07	23.97	14.35	9.24	5.99	19.74
50	17.84	24.05	21.57	9.08	6.56	15.96
51	25.60	21.12	6.11	9.47	11.03	17.45
52	12.73	20.61	28.22	9.81	6.99	15.71
53	6.56	20.65	10.45	9.11	6.99	15.31
54	26.07	20.43	6.07	10.08	13.93	15.46
55	8.24	23.90	7.49	9.72	4.99	15.88
56	44.59	23.36	36.70	9.46	9.01	15.91
57	7.98	22.14	20.45	8.98	10.80	16.05
58	16.23	23.76	13.27	9.13	5.02	15.98
59	26.31	22.38	24.38	10.01	7.99	15.74
60	23.86	22.66	31.91	9.67	7.99	15.75
61	13.16	23.46	21.49	10.11	3.00	17.92
62	12.00	24.32	19.56	10.55	6.04	19.45
63	18.47	23.27	5.26	10.42	10.82	19.59
64	27.21	23.61	18.30	10.34	5.99	19.55
65	9.21	23.40	6.29	10.86	9.53	19.48
66	16.56	23.62	9.70	10.91	7.30	19.00
67	9.16	25.49	4.97	11.72	7.10	18.22
68	21.37	26.51	18.12	11.08	7.00	18.70
69	6.99	25.31	10.07	10.61	9.66	18.69
70	37.98	25.26	18.80	12.01	4.87	18.82
71	36.96	25.34	8.01	11.23	7.45	19.92
72	13.47	24.86	28.71	11.08	3.88	17.88
mean	18.78	23.48	16.26	10.20	7.50	17.59

^aComputational time (second).

Table 20. Average RPD_{avg} and PRE_{avg} errors and computational times for each job group

Job	GA			SA			PSO		
group	Avg PRE_{avg}	Avg RPD_{avg}	Avg CPU_{avg}^a	Avg PRE_{avg}	Avg RPD_{avg}	Avg CPU_{avg}^a	Avg PRE_{avg}	Avg RPD_{avg}	Avg CPU_{avg}^a
6	12.76	*	0.53	11.24	*	0.28	8.08	*	0.39
8	19.63	*	1.29	26.29	*	0.76	16.38	*	1.12
14	*	25.24	8.28	*	23.56	5.16	*	11.94	7.58
16	*	14.82	11.15	*	20.53	5.59	*	9.47	9.37
30	*	19.01	22.42	*	18.42	9.48	*	8.11	16.25
40	*	18.55	24.54	*	14.11	10.91	*	6.89	18.91

^aComputational time (second).

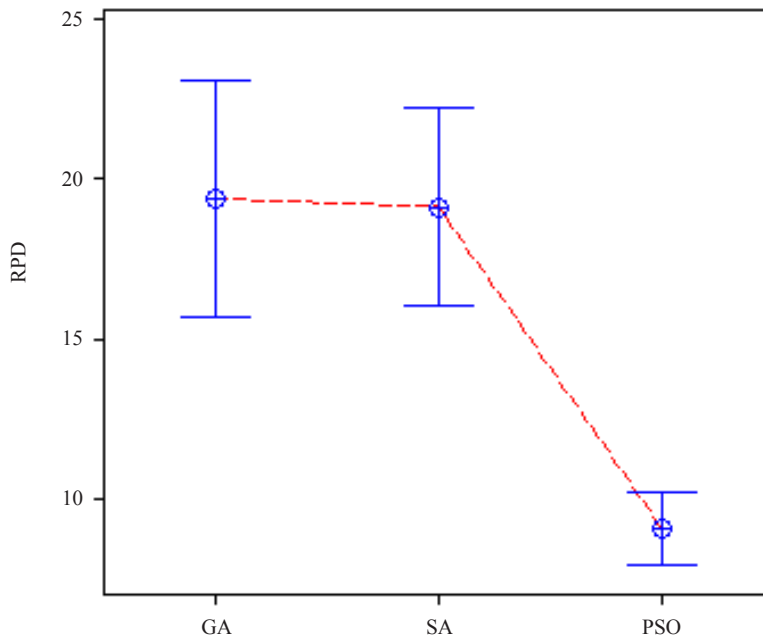


Figure 11. Relative percentage deviation means plot

It can be observed from Figure 12 that by increasing the number of jobs, the quality of PSO and SA solutions enhance while GA does not follow such a constant improvement. Also, it is clear that in each job group, the obtained solution by PSO is completely better than the SA and GA. PSO starts with nearly 12% average RPD_{avg} and becomes less and less until it ends at nearly 7%. In addition, from the obtained results presented in Figure 11, it can be seen that PSO significantly outperforms GA and SA in terms of total E/T. Also, it can be stated that the SA and GA algorithms are not statistically different which means the t -test does not reject the null hypothesis of the equality of SA and GA. So, the results indicate that the performance of the proposed PSO is consistently better than the suggested GA and SA in all conducted experimental tests. Thus, PSO is recommended for solving the proposed problem.

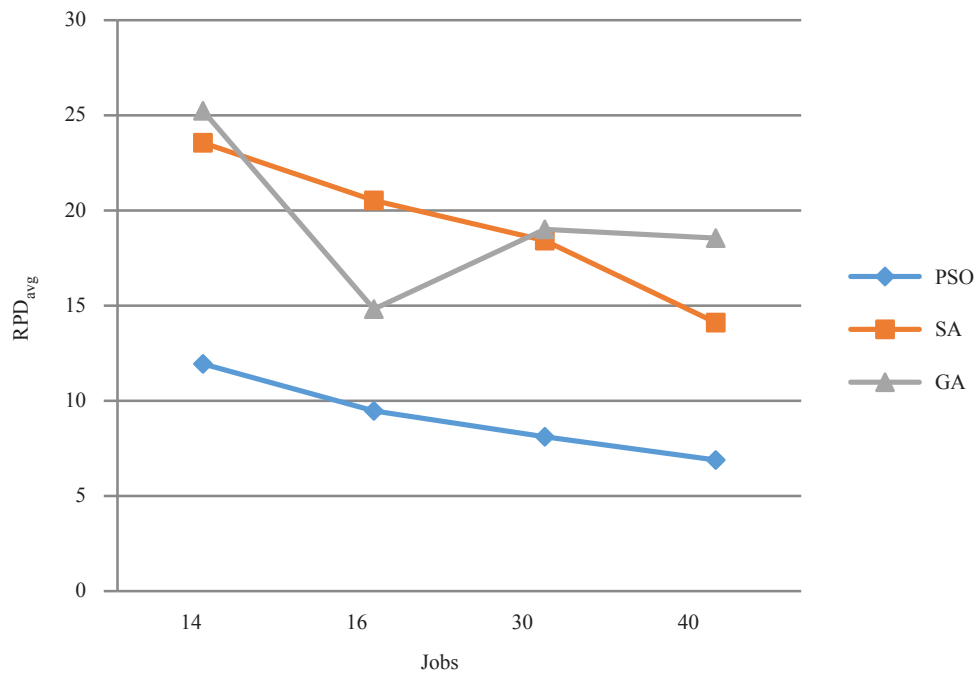


Figure 12. The comparison of the average RPD_{avg} for job groups

6. Conclusions and future work

This paper discussed the problem of scheduling uniform parallel machines with past-sequence-dependent setup time, position-based learning effect, and processing set restrictions to minimize the total earliness/tardiness costs. The idea of the considered problem has arisen from several industries such as the automotive brake systems industry. A new mixed-integer linear programming has been presented to model this problem. Since the problem is strongly NP-hard, the proposed mathematical model just finds optimal solutions for the small-sized problems in a reasonable computational time. Therefore, to find the near-optimal solution for large-sized problems within reasonable times, three meta-heuristic algorithms, genetic algorithm, simulated annealing, and particle swarm optimization algorithm are presented. Also, a heuristic algorithm is proposed for calculating the best JIT objective function based on a given sequence of jobs. The three proposed meta-heuristic algorithms benefit from this exact method. All the proposed algorithms can find optimal solutions for small-sized instances in a very short time. Furthermore, through quite extensive computational experiments, it has been found that the proposed PSO outperforms the other two algorithms, and thus it is recommended as the best solution approach. Developing the model with other objective functions, considering the sum of processing-time-based learning effect instead of position-based learning effect and release date instead of processing set restrictions can be investigated for future research.

Conflict of interest statement

The authors declare no competing financial interest.

References

- [1] Shokoufi K, Rezaeian J. An exact solution approach using a novel concept for single machine preemptive scheduling problem in the just-in-time production system. *Journal of Industrial and Production Engineering*. 2020;

- 37(5): 215-228. Available from: <https://doi.org/10.1080/21681015.2020.1772384>.
- [2] Allahverdi A, Ng CT, Cheng TE, Kovalyov MY. A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*. 2008; 187(3): 985-1032.
 - [3] Ying KC, Lee ZJ, Lin SW. Makespan minimization for scheduling unrelated parallel machines with setup times. *Journal of Intelligent Manufacturing*. 2012; 23(5): 1795-1803.
 - [4] Rezaeian J, Mohammad-Hosseini S, Zabihzadeh S, Shokoufi K. Fuzzy scheduling problem on unrelated parallel machine in JIT production system. *Artificial Intelligence Evolution*. 2020; 1(1): 17-33.
 - [5] Ramezani P, Rabiee M, Jolai F. No-wait flexible flowshop with uniform parallel machines and sequence-dependent setup time: A hybrid meta-heuristic approach. *Journal of Intelligent Manufacturing*. 2015; 26(4): 731-744.
 - [6] Cota LP, Coelho VN, G uimarães FG, Souza MJ. Bi-criteria formulation for green scheduling with unrelated parallel machines with sequence-dependent setup times. *International Transactions in Operational Research*. 2021; 28(2): 996-1017.
 - [7] Ahmadizar F, Mahdavi K, Arkat J. Unrelated parallel machine scheduling with processing constraints and sequence dependent setup times. *Advances in Industrial Engineering*. 2019; 53(1): 495-507.
 - [8] Vallada E, Ruiz R. A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research*. 2011; 211(3): 612-622.
 - [9] Rezaeian J, Zarei M, Shokoufi K. Pareto-based multi-criteria evolutionary algorithm for a parallel machines scheduling problem with sequence-dependent setup times. *International Journal of Engineering*. 2017; 30(12): 1863-1869.
 - [10] Kim HJ, Lee JH. Scheduling uniform parallel dedicated machines with job splitting, sequence-dependent setup times, and multiple servers. *Computers & Operations Research*. 2021; 126: 105115.
 - [11] Biskup D. A state-of-the-art review on scheduling with learning effects. *European Journal of Operational Research*. 2008; 188(2): 315-329.
 - [12] Azzouz A, Ennigrou M, Said LB. Solving flexible job-shop problem with sequence dependent setup time and learning effects using an adaptive genetic algorithm. *International Journal of Computational Intelligence Studies*. 2020; 9(1-2): 18-32.
 - [13] Shokoufi K, Rezaeian J, Shirazi B, Mahdavi I. Preemptive just-in-time scheduling problem on uniform parallel machines with time-dependent learning effect and release dates. *International Journal of Operational Research*. 2019; 3: 339-368.
 - [14] Azadeh A, Goodarzi AH, Kolae MH, Jebreili S. An efficient simulation-neural network-genetic algorithm for flexible flow shops with sequence-dependent setup times, job deterioration and learning effects. *Neural Computing and Applications*. 2019; 31(9): 5327-5341.
 - [15] Kuo WH, Hsu CJ, Yang DL. Some unrelated parallel machine scheduling problems with past-sequence-dependent setup time and learning effects. *Computers & Industrial Engineering*. 2011; 61(1): 179-183.
 - [16] Liao B, Wang X, Zhu X, Yang S, Pardalos PM. Less is more approach for competing groups scheduling with different learning effects. *Journal of Combinatorial Optimization*. 2020; 39(1): 33-54.
 - [17] Li D, Wang J, Qiang R, Chiong R. A hybrid differential evolution algorithm for parallel machine scheduling of lace dyeing considering colour families, sequence-dependent setup and machine eligibility. *International Journal of Production Research*. 2021; 59(9): 2722-2738.
 - [18] Huo Y, Leung JYT. Parallel machine scheduling with nested processing set restrictions. *European Journal of Operational Research*. 2010; 204(2): 229-236.
 - [19] Gokhale R, Mathirajan M. Scheduling identical parallel machines with machine eligibility restrictions to minimize total weighted flowtime in automobile gear manufacturing. *The International Journal of Advanced Manufacturing Technology*. 2012; 60(9): 1099-1110.
 - [20] Perez-Gonzalez P, Fernandez-Viagas V, García MZ, Framinan JM. Constructive heuristics for the unrelated parallel machines scheduling problem with machine eligibility and setup times. *Computers & Industrial Engineering*. 2019; 131: 131-145.
 - [21] Rezaeian J, Derakhshan N, Mahdavi I, Alizadeh Foroutan R. Due date assignment and JIT scheduling problem in blocking hybrid flow shop robotic cells with multiple robots and batch delivery cost. *International Journal of Industrial Mathematics*. 2021; 13(2): 145-162.
 - [22] Baker KR, Trietsch D. *Principles of sequencing and scheduling*. John Wiley & Sons; 2013.
 - [23] Graham RL, Lawler EL, Lenstra JK, Kan AR. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*. 1979; 5: 287-326.
 - [24] Hall NG, Posner ME. Earliness-tardiness scheduling problems, I: weighted deviation of completion times about a common due date. *Operations Research*. 1991; 39(5): 836-846.

- [25] Biskup D. Single-machine scheduling with learning considerations. *European Journal of Operational Research*. 1999; 115(1): 173-178.
- [26] Holland JH. Adaptation in natural and artificial systems. University of Michigan, Ann Arbor; 1975.
- [27] Henderson D, Jacobson SH, Johnson AW. The theory and practice of simulated annealing. *Handbook of metaheuristics*. Springer, Boston, MA; 2003. p. 287-319.
- [28] Kennedy J, Eberhart R. Particle swarm optimization. *Proceedings of ICNN'95-international conference on neural networks*. IEEE; 1995. p. 1942-1948.
- [29] Potts CN, Van Wassenhove LN. A decomposition algorithm for the single machine total tardiness problem. *Operations Research Letters*. 1982; 1(5): 177-181.