International Journal of Industrial Engineering & Production Research (2019) 30: 329-340
DOI: 10.22068/ijiepr.30.3. 329

RESEARCH PAPER

A Job Shop Scheduling Problem with Sequence-Dependent Setup Times Considering Position-Based Learning effects and Availability Constraints

Seyyedhamed Mousavipour[1], Hiwa Farughi[2*]& Fardin Ahmadizar[3]

Received 13 August 2018; Revised 30 April 2019; Accepted 13 May 2019; Published online 16 September 2019

ABSTRACT

Sequence-dependent setup times (SDSTs) scheduling problems, learning effects, transportation times, and availability constraints are significant and appealing issues in production management. Researchers often study these issues in isolation, and these constraints have rarely been considered together. The present paper investigated the SDSTs job shop scheduling problem (JSSP) with position-based learning effects, job-dependent transportation times, and multiple preventive maintenance activities such that, as a result of the learning effects, the times required for processing the jobs were variable during the planning horizon, and each machine had a predetermined number of preventive maintenance activities. For the formulation of the problem in order to minimize makespan, a new mixed-integer linear programming (MILP) model was proposed. Since the problem was highly complex, Grey Wolf Optimizer (GWO) and Invasive Weed Optimizer (IWO) were employed so that near-optimal solutions to medium- and large-sized instances could be obtained. To evaluate the performance and effectiveness of the proposed solution methods, the computational results were used.

KEYWORDS: *Sequence-dependent setup times, Job shop scheduling problem, position-based learning effects, Availability constraints, Transportation times.*

1. Introduction

A JSSP is addressed in the current paper in the presence of sequence-dependent setup times. JSSP, which was introduced by Muth and Thompson [1], is one of the most important scheduling problems. In a standard job shop system, a set of *n* jobs should be processed by a set of *m* machines. A distinct routine is assigned to each job so that the corresponding operations can be processed by a given set of machines. Recently, most scholars have attempted to provide mathematical models closer to the real world by considering a number of parameters and assumptions such as sequence-dependent setup times, transportation times, learning effects, and availability constraints.

In the literature, most of the researchers have supposed that setup times are either insignificant or included in processing times. However, Allahverdi and Soroush [2] stated that scheduling activities were dependent on the preparation of related facilities and instrument, and scheduling with setup times played a crucial role in this context.

Operations can be performed simultaneously, and resource utilization is improved once setup times and processing times are considered separately. Therefore, it is highly necessary to emphasize the significance, advantages, and applications of the explicit consideration of setup times in research on scheduling [3].

A distinction should be made between sequence-independent setup times and sequence-dependent ones. In the case of sequence-independent setup times, it is only the job that should be processed depending on setup, whereas for sequence-dependent setup times (known as SDSTs), setup depends on the job that has been completed and on the one that should be processed. There are two types of SDSTs: anticipatory and Non-anticipatory [2]. A non-anticipatory setup is the one that starts only when the corresponding job and the corresponding machine are both available, while an anticipatory setup can start

* Corresponding author: *Hiwa Farughi*
h.farughi@uok.ac.ir

1. *Department of Industrial Engineering, University of Kurdistan, Sanandaj, Iran.*
2. *Department of Industrial Engineering, University of Kurdistan, Sanandaj, Iran.*
3. *Department of Industrial Engineering, University of Kurdistan, Sanandaj, Iran.*

once the machine is available even where the job that should be processed is unavailable.

As in the cases of other manufacturing environments, the first line of research conducted on SDSTs is concerned with single machines. Coleman [4] proposed an integer programming model for minimizing earliness and tardiness in a single machine assuming sequence-dependent setup times. A large number of researchers then made plenty of investigations in that area. A comprehensive survey of research on scheduling involving setup times can be found in Allahverdi et al. [5], Zandieh et al. [6], and Allahverdi et al. [7].

As for SDSTJSSP, Choi and Korkmaz [8] addressed the two-job/*m*-machine with anticipatory SDSTs and release times. They provided a mixed-integer programming model for the problem and proposed a heuristic method based on the consecutive recognition of a pair of operations, which provide a minimal lower bound on makespan. Schutten [9] studied a practical job shop environment with respect to a number of conditions such as release and due dates, setup times, and transportation times. He did not provide any mathematical model, but proposed an extension to the shifting bottleneck procedure for solving the problem. Naderi et al. [10] investigated SDSTJSSP under conditions of makespan minimization. They did not provide any mathematical model either for solving the problem, but developed an effective metaheuristic based on simulated annealing with novel operators. Shen [3] studied classical SDSTJSSP. He proposed a mathematical model based on a modified disjunctive graph. For solving the model, a Tabu search algorithm with a sophisticated neighborhood structure was developed. Ahmadizar and Shahmaleki [11] provided a mathematical model for a group shop scheduling problem with SDSTs and transportation times. They proposed a genetic algorithm (GA) hybridized with an active schedule generator to tackle large-sized instances.

In the job shop environment, jobs move between machines. In the past, most studies assumed that transportation times were negligible. In the real world, however, a job may not often be processed on a machine immediately after the completion of its preceding operation because of the transportation times. According to Naderi et al. [12], transportation times can be of either of two types: job-independent and job-dependent. It is only the distance between the two consecutive machines or stages that the magnitude of a transportation time depends on in the job-independent type. In the job-dependent type, however, the job that should be carried out is also considered for the specification of magnitude. Furthermore, there are single-transporter and multi-transporter alternatives for the transportation system [13]. A multi-transporter system is characterized by several (unlimited) transporters that are there for performing jobs; in such a system, therefore, jobs never have to wait for a transporter before transported. In a single-transporter system, on the other hand, a single transporter is used for carrying out all inter-stage transportations; jobs should await the transporter to return. Where the SDSTs and transportation times are considered simultaneously, two cases need to be considered. The setups and transportations may overlap in one case corresponding to anticipatory SDSTs, and a job can, therefore, begin being processed by a machine provided that the required transportation and setup are completed. In another case associated with non-anticipatory SDSTs, the setups and transportations may not overlap, i.e., the setup of a machine cannot begin before the transportation is completed [11]. A generalized JSSP was addressed by Hurink and Knust [13], where a single-transport robot was used to transport the jobs between the machines. The empty movement times for the robot and the transportation times for the jobs were considered. The objective is to minimize makespan in the specified schedule. A Job Shop Scheduling Problem with Transportation times and Many Robots (JSPT-MR) was investigated by Nouri et al. [14]. A hybrid metaheuristic approach based on the clustered holonic multi-agent model for the JSPT-MR is proposed in the present paper.

Another assumption found in most scheduling studies is that production resources, including machines, are always available, while this assumption is violated in the real world for reasons such as machine failure and preventive maintenance activities, tool change and overlapped planning horizons [15]. Schmidt [16] and Ma [17] conducted a comprehensive review of scheduling models with the availability constraints. In this paper, as in the case of Vahedi-Nouri et al. [18], a predetermined machine availability constraint is considered. One or more preventive maintenance activities are performed by each machine during its operation so that its lifespan is prolonged, and breakdown is minimized.

According to the learning effects, production facility performance improves gradually as time passes. Consequently, the later a given job is

scheduled in the sequence, the shorter the processing time it will have [19].

The practical applications and results of learning effects on productivity have been observed in many industries including those in the manufacturing and service sectors. For example, in assembly workshops, such as assembly lines of cars, ships, aircraft, and so on, wherever an operator does a job repeatedly, it generally gains knowledge and experience each time. Consequently, the processing times of jobs are reduced at each repetition. Biskup [20] was the first researcher who considered the concept of learning effects in scheduling problems. Afterward, Biskup [21] conducted a comprehensive study of the various types of learning effects in different scheduling environments. He divided learning effects into two general types: position-based learning effects and learning effects based on the sum of processing times. In the former category, learning corresponds to a number of jobs processed so far, while, in the second one, learning is in accordance with the sum of the processing times of the jobs processed so far. Many studies have investigated the various types of learning effects in different scheduling environments; Amirian and Sahraeian [22], Cheng et al. [23], Salehi and Rezaeian [24], and Okolowski and Gawiejnowicz, [25] are among them. This paper provides a position-based MILP model for SDSTJSSP based on assumptions that have rarely been taken into account in this problem so far.

The rest of this paper is organized as follows. In Section 2, a novel mixed-integer linear programming model for the JSSP is provided, given the classical position-based learning effects on the processing times of jobs, anticipatory sequence-dependent setup times, preventive maintenance, and job-dependent transportation times. Model validation is discussed in Section 3. In Section 4, the metaheuristic solution techniques utilized for approaching the model are detailed. The findings obtained by the algorithms are evaluated and discussed in Section 5. Finally, the conclusion is drawn in Section 6.

2. Problem Description

In this section, the JSSP is described with the aforementioned assumptions. A set of n jobs $\{j_1, j_2, \ldots, j_n\}$, each containing L operations, must be processed in a predetermined sequence on a set of m machines $\{M_1, M_2, \ldots, M_m\}$. Each job can be processed only by one machine at a time, and each machine can process only one job at a time. Pre-emption is not allowed, all the jobs are available at time zero, and no inter-machine buffer limitation is there. Every job J_j is processed in a known, fixed normal processing time of P_{ij}. It should be noted that the learning effects influence the jobs in such a way that the actual processing time of job J_j on machine M_i is dependent upon the position that the job has in the sequence and on the learning index $a_i \leq 0$ such that $p_{ijk} = p_{ij} \times k^{a_i}$.

Anticipatory sequence-dependent setup times are considered, and $S_{jij'}$ indicates the time needed for the preparation of machine M_i where j_j is processed on that machine prior to $j_{j'}$. An additional assumption is made that $S_{0ij} = 0$.

Moreover, it has been assumed in the literature that a triangular inequality is met by sequence-dependent setup times; hence, $S_{jij'} + S_{j'ij''} \geq S_{jij''}$ where this seems like a rational condition based on the practical applications [11]. In the case of transportation times, an unlimited multi-transporter system is considered that contains job-dependent transportation times $tp_{iji'}$ representing the time needed for transferring job J_j from machine M_i to machine $M_{i'}$. Along the same lines as the setup times, the transportation times are to meet the triangle inequality, i.e., $tp_{iji'} + tp_{i'ji''} \geq tp_{iji''}$; this is, of course, not a real restriction. On the other hand, machine M_i needs r maintenance activities, so that the r^{th} maintenance activity PM_{ir} with an execution time of t_{ir} is performed after the predefined number of jobs is processed.

The aim is to find the sequence of jobs processed on machines in such a way that makespan is minimized. C_{ijk} is the completion time of job J_j if it is scheduled in the k^{th} position of machine M_i, and C_j is the completion time of job J_j. The binary variable X_{ijk} is 1 if job J_j is scheduled in the k^{th} position of machine M_i, and is 0 otherwise. The variable $Z_{ijkj'k+1} = X_{ijk} \times X_{ij'k+1}$ is 1 if job J_j is processed in the k^{th} position of machine M_i, and job $J_{j'}$ is processed in the k+1th position of the machine. y_{irk} is a binary parameter, which is 1 if the r^{th} maintenance activity of machine M_i is performed after the job in the k^{th} position of the machine is processed, and is 0 otherwise. Finally, V is a very large number. Notations, decision variables briefly and finally the mathematical model are defined as follows:

1-2. Notations

1-1-2. Indices:

i: Machine index

j: Job index
k: Position index
r: Maintenance index
L: Operation index

1-2-2. Parameters:

n: Number of jobs
m: Number of machines
V: A large positive number
r: Number of maintenance activities performed on each machine
P_{ij}: Normal processing time of job J_j on machine M_i
P_{ijk}: Processing time of job J_j on the k^{th} position of machine M_i
α_i: Job processing learning index on machine M_i ($\alpha_i \leq 0$)
PM_{ir}: r^{th} maintenance activity on machine M_i
t_{ir}: Runtime of PM_{ir}
r_{ijl}: A binary parameter that is 1 if the l^{th} operation of the j^{th} job is processed on machineM_i, and is 0 otherwise
$tp_{iji'}$: Time needed for the transfer of job J_j from machine M_i to machine $M_{i'}$
$S_{jij'}$: Setup time of machine M_i, when job J_j is processed by this machine before $J_{j'}$
y_{irk}: A binary parameter that is 1 if the r^{th} maintenance activity must be done after processing the job on the k^{th} position of machine M_i and 0 otherwise

2-2. Decision variables

X_{ijk}: A binary variable that is 1 if job J_j is processed on the k^{th} position of machine M_i, and is 0 otherwise
C_{ijk}: Completion time of job J_j if scheduled on the k^{th} position of machine M_i and 0 otherwise
C_{max}: Makespan
$Z_{ijkj'k+1} = X_{ijk} \times X_{ij'k+1}$ is 1 if job J_j is processed in the k^{th} position of machine M_i, and job $J_{j'}$ is processed in the k+1$^{\text{th}}$ position of the machine

2-3. Mathematical model

$$\text{M}in\ Z = C_{max} \tag{1}$$

s.t

$$\sum_{\text{k}} \text{X}_{\text{ijk}} = 1 \qquad \forall \text{ i} = 1, \dots, \text{m};\ \text{j} = 1, \dots, \text{n} \tag{2}$$

$$\sum_{\text{j}} \text{X}_{\text{ijk}} = 1 \qquad \forall \text{ i} = 1, \dots, \text{m};\ \text{k} = 1, \dots, \text{n} \tag{3}$$

$$\sum_{\text{j}} \text{C}_{\text{ijk}} + \sum_{\text{j}} \text{X}_{\text{ij k+1}} \times \text{P}_{\text{ij k+1}} + \sum_{r=1}^{r} y_{irk} \times t_{ir} + \sum_{\text{j}} \sum_{j'} Z_{ijkj'k+1} \times \text{S}_{\text{jií}} \leq \sum_{\text{j}} C_{ijk+1} \qquad \forall \text{ i} = 1, \dots, \text{m};\ \text{k} = 1, \dots, \text{n} - 1 \tag{4}$$

$$\sum_{\text{i}} \text{r}_{\text{ijl}} \times \text{C}_{\text{ijk}} + \sum_{\text{í}} \text{r}_{\text{íj l+1}} \times \text{X}_{\text{íjk}} \times \text{P}_{\text{íjk}} + \sum_{\text{i}} \sum_{\text{í}} \text{r}_{\text{ijl}} \times \text{r}_{\text{íj l+1}} \times \text{tp}_{\text{ijí}} \leq \text{V} \times \left(1 - \sum_{\text{i}} \text{r}_{\text{ijl}} \times \text{X}_{\text{ijk}}\right) + \text{V} \times \left(1 - \sum_{\text{j}} \text{r}_{\text{ij l+1}} \times \text{X}_{\text{ijk}}\right) + \sum_{\text{í}} \text{r}_{\text{íj l+1}} \times C_{\text{íjk}} \qquad \forall j = 1, \dots, n;\ L = 1, \dots, m\text{-}1;\ K = 1, \dots, n;\ \text{í}, i = 1, \dots, \text{m} \tag{5}$$

$$Z_{ijkj'k+1} \leq X_{i,j,k} \qquad \forall i = 1, \dots, m;\ j, j' = 1, \dots, n;\ \text{k} = 1, \dots, \text{n} - 1 \tag{6}$$

$$Z_{ijkj'k+1} \leq X_{ij'k+1} \qquad \forall i = 1, \dots, m;\ j, j' = 1, \dots, n;\ \text{k} = 1, \dots, \text{n} - 1 \tag{7}$$

$$Z_{ijkj'k+1} \geq \left(X_{ijk} + X_{ij'k+1}\right) - 1 \qquad \forall i = 1, \dots, m;\ j, j' = 1, \dots, n;\ \text{k} = 1, \dots, \text{n} - 1 \tag{8}$$

$$C_{ijk} \leq V \times X_{ijk} \qquad \forall \text{ i} = 1, \dots, \text{m};\ j = 1, \dots, n;\ \text{k} = 1, \dots, \text{n} \tag{9}$$

$$\sum_{k} C_{[1]jk} \geq \sum_{k} X_{[1]jk} \times P_{[1]jk} \qquad \forall \text{ j} = 1, \dots, \text{n} \tag{10}$$

$$C_j \geq \sum_k C_{[Last\ machine]jk} \qquad \forall j = 1, \dots, n \qquad (11)$$

$$C_{max} \geq C_j \qquad \forall j = 1, \dots, n \qquad (12)$$

$$C_j \geq 0 \qquad \forall j = 1, \dots, n \qquad (13)$$

$$X_{ijk}, Z_{ijkj'k+1}, \in \{0,1\} \quad \forall i = 1, \dots, m \quad k = 1, \dots, n \quad r = 1, \dots, r, \quad j, j' = 1, \dots, n \qquad (14)$$

In this model, the first equation represents the objective function, i.e., minimization of the maximum completion times of jobs. According to Constraints (1) and (2), each job can be assigned only to one position, and each position can involve only one job. Constraint (3) computes the completion time of the job on each machine according to the job processing times, maintenance activity times, SDSTs, and learning effects. Constraint (5) is the precedence constraint. It ensures that all the operations of a job are executed in the given order. Constraints (6), (7), and (8) are applied to the linearization equation $Z_{ijkj'k+1} = X_{ijk} . X_{ij'k+1}$. Equation (9) states that the completion time of job J_j is 1 if it is scheduled in the k^{th} position of machine M_i, and is 0 otherwise. Constraints (10)-(11) calculate the completion time of each job and makespan. Constraints (13) and (14) determine positive and binary variables.

3. Model Validation

Because of the originality of the model and the lack of benchmarks, the model is validated using the Gantt chart. The model is solved for each machine in a small-sized instance with 2 jobs, 3 machines, and 1 maintenance activity to minimize makespan. Table 1 gives the sequence of jobs processed on machines, the normal processing time of each job on each machine, duration of maintenance, inter-machine transportation times, and sequence-dependent setup times.

Moreover, Table 2 shows the job processing times affected by position-based learning effects. It is worth mentioning that, for simplicity, the same learning rates have been considered for both machines.

Tab. 1. Input parameters for instances with 3 machines and 2 jobs

Normal processing times			Sequence			
	Job1	Job 2	Job 1 (M_1-M_2-M_3)			
M_1	10	15	Job 2 (M_3-M_2-M_1)			
M_2	18	9				
M_3	5	12				
Maintenance matrix			Duration of maintenance			
	Position 1					
M_1-1	1		t_{11}	3		
M_2-1	1		t_{21}	3.5		
M_3-1	1		t_{31}	1.5		
Transportation matrix				M_1	M_2	M_3
	$tp_{iji'}$		M_1-J_1	0	4	5
			M_1-J_2	0	3	4
			M_2-J_1	2	0	6
			M_2-J_2	3.5	0	4
			M_3-J_1	1.3	4.5	0
			M_3-J_2	3	2	1
Sequence-dependent set-up times				J1	J2	
	$S_{jij'}$		J_1-M_1	0	5	
			J_1-M_2	0	3	
			J_1-M_3	0	2	
			J_2-M_1	9	0	
			J_2-M_2	3	0	
			J_2-M_3	0	0	

Tab. 2. Actual processing times of jobs

Actual job processing times		Position 1	Position 2
p_{ijk}	M_1-J_1	10	8.12
Learning index α_i = -0.3	M_1-J_2	15	12.18
	M_2-J_1	18	13.17
	M_2-J_2	9	6.58
	M_3-J_1	5	3.41
	M_3-J_2	12	8.19

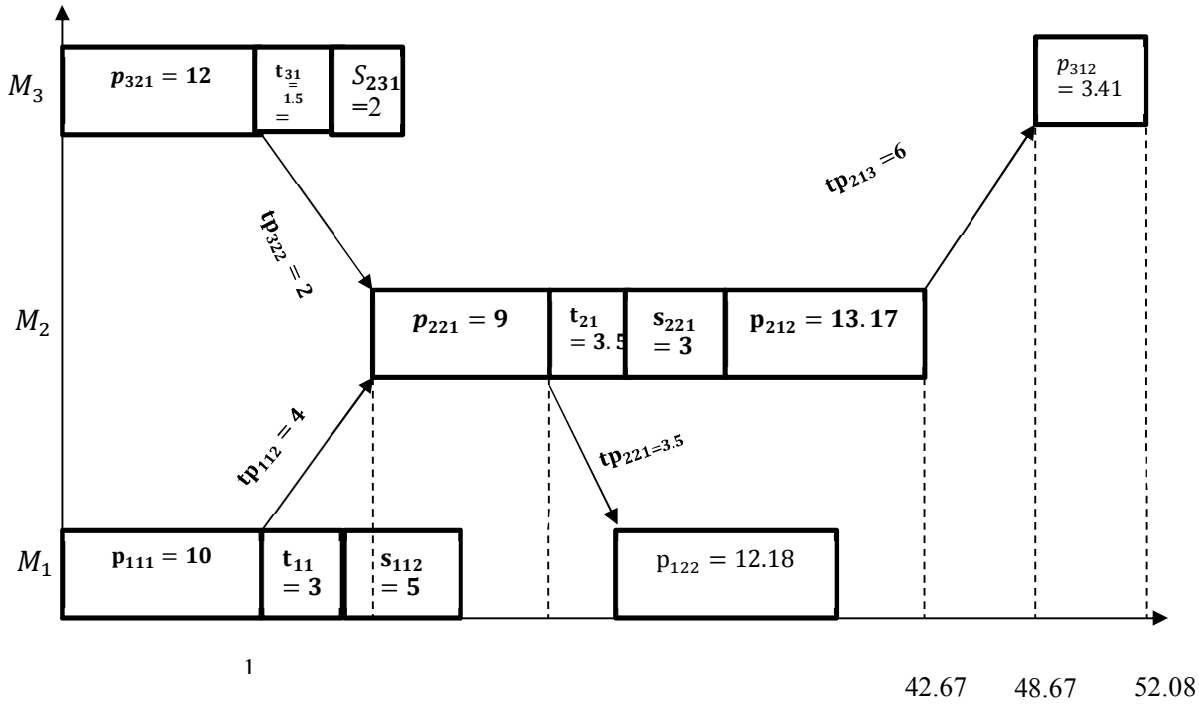

Fig. 1. Gantt chart for the optimal solution of an instance with 3 machines and 2 jobs

According to the Gantt chart (Fig. 1), the optimal value for the objective function of the problem is 52.08. It is worth mentioning that there is no overlap between the maintenance activities and setup operations, and the last maintenance activity on machine M_i has no effect on the completion time of the last job on that machine, and only the maintenance activity between the processing of two consecutive jobs affects the value of the objective function. Other feasible solutions to the problem and their corresponding makespan values are as follows. The optimal makespan values achieved by the Gantt chart and the proposed model are equal, demonstrating the validity of the model.

$X_{111} = X_{122} = X_{212} = X_{221} = X_{312} = X_{321} = 1$ CMAX = 52.08

$X_{111} = X_{122} = X_{212} = X_{222} = X_{321} = X_{312} = 1$ CMAX = 74.77

$X_{121} = X_{112} = X_{221} = X_{212} = X_{321} = X_{312} = 1$ CMAX = 83.61

$X_{121} = X_{112} = X_{211} = X_{222} = X_{311} = X_{322} = 1$ CMAX = 77.45.

4. Metaheuristic Solution Method

JSSP is a NP-hard problem [26]. The learning effects on the parameters of the problem add a dimension to it, making the problem more complicated; therefore, for solving a large-sized problem, it is critical to utilize appropriate and efficient solution methods. Solution methods are proposed based on the Grey Wolf and Invasive Weed algorithms. The solution is presented as a string of decimals. The solution representation should represent only one solution to the problem. The length of this numeric string is equal to *the number of machines* × *the number of jobs*. Each string is divided into sections as many as the number of machines, and the numbers in

each section are arranged in ascending order. In each section, the order of the numbers arranged for the corresponding machine determines the order of the jobs assigned to that machine. In other words, these are feasible solutions. Figs. 2 and 3 demonstrate the solution representations.

۰,۸۲۳۶۶۵	۰,۷۰۵۷۴۵	۰,۳۱۳۷۷۶	۰,۱۶۱۵۸	۰,۳۱۶۶۴۹	۰,۷۹۲۴۳۱	۰,۳۱۴۶۶۰۹	۰,۸۸۳۷۵۳	۰,۱۶۰۴۸۶

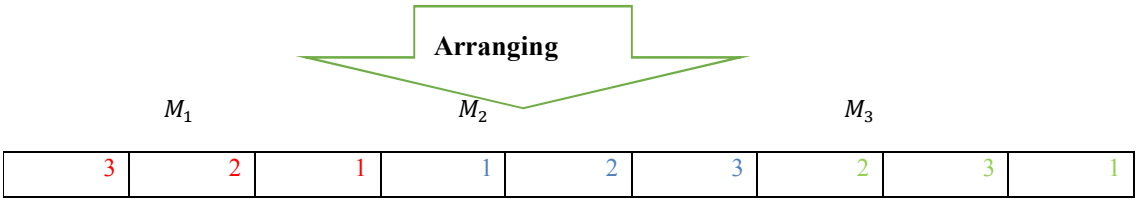

Fig. 2. Solution representation

x_{1jk}

0	0	1
0	1	0
1	0	0

x_{2jk}

1	0	0
0	1	0
0	0	1

x_{3jk}

0	1	0
0	0	1
1	0	0

Fig. 3. Representation of a feasible solution

In order to gain feasible solutions, an initial solution is generated for the problem at the beginning of the algorithm with a heuristic method that generates the first feasible solution for each machine greedily. Fig.4 depicts this procedure.

The representation of the problem solution should be simulated to calculate the values of the objective function of the problem. Therefore, the completion time value of each job is computed given the problem constraints.

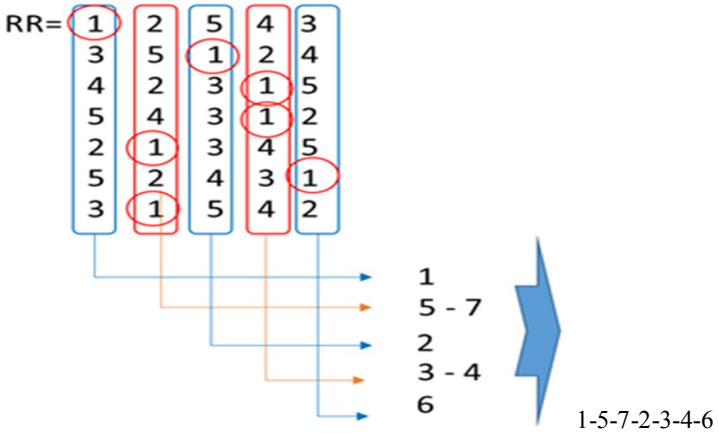

Fig. 4. Generating an initial solution

1-4. Grey wolf optimizer (GWO)

Inspired by grey wolves (Canis lupus), a novel metaheuristic referred to as Grey Wolf Optimizer (GWO) was proposed by Mirjalili and Mirjalili [27]. According to them, the algorithm imitates the natural hunting mechanism and leadership hierarchy of grey wolves. For simulating the latter, four types of grey wolf are used: alpha, beta, delta, and omega. Moreover, the three major stages involved in the hunting strategy are implemented. These include searching for, encircling, and attacking prey. The pseudo-code of the algorithm is shown in Fig. 5. This method was comprehensively described in Ref [27].

2-4. Invasive weed optimizer (IWO)

Inspired by colonizing weeds, a new numerical stochastic optimization algorithm was introduced by Mehrabian and locus [28]. According to them, weeds constitute a type of plant that threatens

agriculture due to its invasive, robust habits of growth, which seriously threatens cultivated, desirable plants. They have demonstrated plenty of vigor and adaptation to environmental changes. Therefore, a highly capable optimization algorithm would result from capturing their properties. A simple, though effective, optimization algorithm, namely Invasive Weed Optimization (IWO), seeks to imitate the vigor, adaptation, and randomness of colonizing weeds. Ref [28] detailed the process involved in IWO.

```
Initialize the population of grey wolves X_i (i = 1, 2, …, n)
Initialize α, A, and C
Calculate the fitness value of each of the search agents
X_α = best search agent
X_β = Second best search agent
X_δ = Third best search agent
While (t < maximum number of iterations)
    For each search agent
        Update the current search agent position
    End for
    Update α, A, and C
    Compute the fitness values of all the search agents
  Update X_α, X_β, and X_δ
  t = t + 1
End While
Return X_α
```

Fig. 5. Pseudo-code of GWO [27]

5. Computational Results

Several small-, medium-, and large-sized instances were generated randomly for testing and analyzing the validity and efficiency of the proposed mathematical model. It is shown in Table 3 how data were generated in these instances. The small-sized instances were solved completely with the CPLEX solver 24.8.5 of GAMS. For the medium- and large-sized instances, however, IWO and GWO were utilized, as GAMS failed to obtain optimal solutions within a reasonable amount of time. Tables 4-6 show the results. It should be noted that parameters of these algorithms were calibrated by trial and error, and MATLAB 2013 was the environment in which the algorithms were coded by a PC equipped with a Core i5, 2.5 GHZ CPU and 4-GB RAM.

Tab. 3. Data generation for random instances

Notation	Parameter	Value
n	Number of jobs	{2…10}
m	Number of machines	{3…20}
k	Number of maintenance activities on each machine	{1…6}
α_i	Learning index of machine M_i	Uniform distribution (-0.3, -0.5)
t_{ir}	Normal maintenance execution time	Uniform distribution (5, 20)
y_{irk}	Maintenance position	Uniform distribution {1, 2, …, n - 1}
$tp_{iji'}$	Transportation times	Uniform distribution (1, 30)
$S_{jij'}$	Setup times	Uniform distribution (5, 40)
P_{ij}	Processing time of job J_j on machine M_i	Uniform distribution (10, 100)

1-5. Performance comparison of the proposed solution methods

The results obtained from solving the instances are shown in Tables 4-6. For a comparison of the solution methods, the value obtained from each approach is transformed into the Relative Percentage Deviation (RPD) using Eq. (15), where Alg_{sol} is the objective value gained by solving an instance using the considered algorithm, and Min_{sol} is the minimum objective value obtained by solving that instance using the solution method. Furthermore, the values in the *Imp* indicate the degree of improvement made in the initial solution, obtained by Eq. (16), where $Alg_{initialsol}$ is the objective value of the initial solution of the algorithm, and $Alg_{finalsol}$ is the objective value of its final solution.

$$\text{RPD} = \frac{Alg_{sol} - Min_{sol}}{Min_{sol}} \times 100 \qquad (15)$$

$$\text{Imp} = \frac{Alg_{initialsol} - Alg_{finalsol}}{Alg_{initialsol}} \times 100 \qquad (16)$$

Tab. 4. Computational results for the small instances

		GWO			IWO			GAMS
Sample	Representation	CPU time (s)	Imp	RPD	CPU time (s)	Imp	RPD	RPD
1	$3 \times 2 \times 1$	12.96	0	0	3.096	0	0	0
2	$4 \times 4 \times 2$	41.96	20%	0	18.67	28%	0	0
3	$4 \times 5 \times 2$	55.02	39%	0	20.48	36%	0	0
4	$3 \times 5 \times 2$	63	0	0	22.43	0	0	0
5	$5 \times 6 \times 2$	70.56	22%	0	46.73	46%	0	0
	Mean	48.70	16.2%	0	22.28	22%	0	0

Tab. 5. Computational results for the medium instances

		GWO			IWO		
Sample	Representation	CPU time (s)	Imp	RPD	CPU time (s)	Imp	RPD
6	$6 \times 5 \times 2$	74.39	18%	0	56.272	44%	4%
7	$6 \times 7 \times 2$	89.087	10%	6%	69.67	48%	0
8	$7 \times 5 \times 2$	83.7.0	13%	0	66..04	7%	9%
9	$7 \times 6 \times 2$	105.33	11%	0	75.12	11%	0
10	$8 \times 5 \times 2$	109.04	1%	3%	77.35	19%	0
11	$8 \times 6 \times 2$	133.65	13%	0	80.45	18%	1%
12	$10 \times 5 \times 2$	156.34	7%	0	90.73	14%	3%
	Mean	107.35	10.42%	1.28%	73.66	23%	2.42%

Tab. 6. Computational results for the large instances

		GWO			IWO		
Sample	Representation	CPU time (s)	Imp	RPD	CPU time (s)	Imp	RPD
13	$12 \times 6 \times 3$	201.2	0	7%	124.85	8%	0
14	$13 \times 6 \times 3$	210.07	15%	0	129.36	20%	2%
15	$14 \times 7 \times 3$	253	18%	0	206.57	30%	10%
16	$15 \times 8 \times 3$	280.07	6%	0	225.65	8%	12%
17	$16 \times 7 \times 4$	292.44	3%	0	231.71	8%	7%
18	$17 \times 8 \times 4$	370.10	2%	0	518	8%	5%
19	$18 \times 8 \times 6$	468	19%	0	330.68	16%	7%
20	$20 \times 10 \times 6$	662.19	3%	0	552.30	12%	6%
	Mean	342.13	8.25%	0.0087%	289.89	13.75%	6.12%

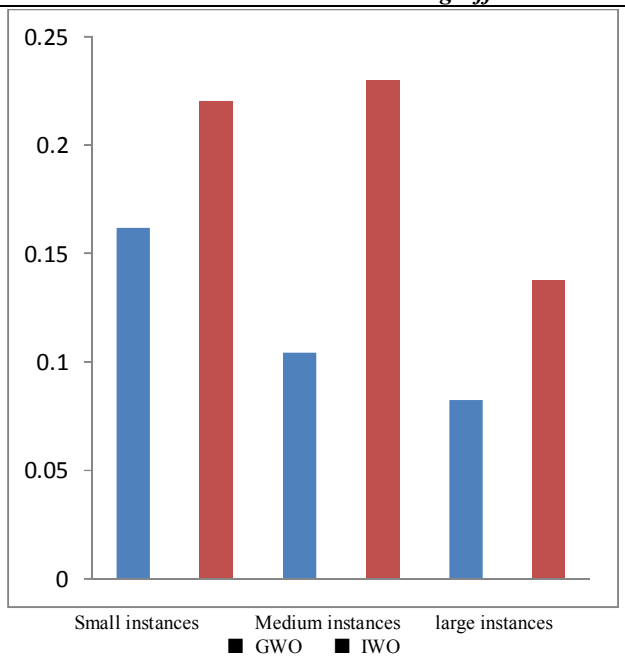

Fig. 6. Mean values of Imp for GWO and IWO

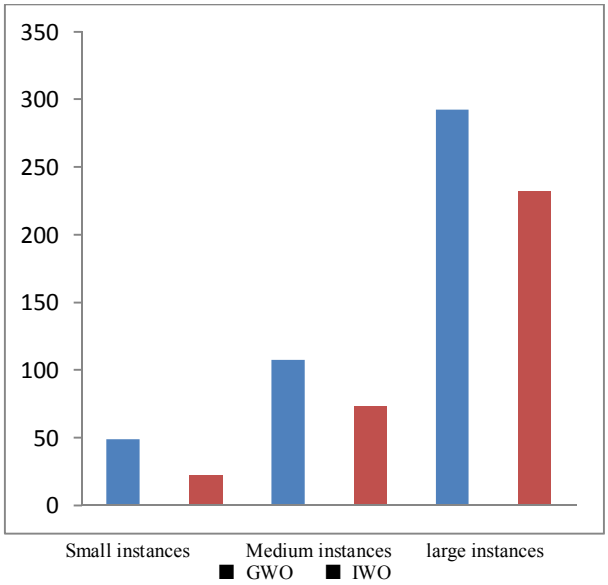

Fig. 7. Mean CPU times of GWO and IWO

According to the RPD values in the small instances, GWO, IWO, and GAMS obtain the same solutions; consequently, both metaheuristics can obtain optimal solutions for small-sized instances. Given the mean RPD values, the performance of GWO as compared to IWO is obviously better in the case of larger problem sizes. As Fig. 6 demonstrates, the improvement made in the initial solution is greater for IWO than that for the GWO algorithm. Furthermore, the great improvement made in all the methods showed they could effectively enhance the initial solution. The mean computation (CPU) times of the two proposed algorithms are compared to one another in all sizes problems, as shown in Fig. 7. As can be deduced from the figure, IWO has a better computation time than the GWO method. Moreover, as the problem scale increases, the average computation time of both methods grows as a polynomial.

6. Conclusion

In this paper, a practical job shop scheduling problem considering sequence-dependent setup

times, position-based learning effects, job-dependent transportation times, and multiple preventive maintenance activities was addressed. A novel MILP model was proposed to minimize makespan. Since the JSSP was a complicated problem, Grey Wolf Optimizer and Invasive Weed Optimizer were utilized for solving the medium and large instances. Since the model was original and there were no benchmarks for the problem, random instances were generated for the evaluation of the behavior and performance of the presented solution methods. The computational results obtained by the proposed algorithms were compared in terms of three criteria and for instances of three sizes. Regarding the RPD values in the small instances, GWO, IWO, and GAMS obtained the same solutions, and the validity of the metaheuristics was confirmed. Given the mean RPD values, the performance of GWO as compared to IWO is obviously better for larger problem sizes. The improvement made in the initial solution for IWO is greater than that for GWO. Moreover, the great improvement made in all the methods reveals their competent performance and effectiveness in the improvement of the initial solution.

Considering other types of availability constraint, such as breakdowns and flexible maintenance, and other types of learning effects and considering non-anticipatory setup times can be appealing suggestions for future studies.

References

[1] Muth, JF. and Thompson, GL. Industrial Scheduling. Englewood Cliffs, New Jersey, (1963).

[2] Allahverdi, A., and Soroush, H. The significance of reducing set-up times/set-up costs. European Journal of Operational Research, Vol. 187, No. 3, (2008), pp. 978-984.

[3] Shen, L. A tabu search algorithm for the job shop problem with sequence dependent set-up times. Computers & Industrial Engineering, Vol. 78, (2014), pp.95-106.

[4] Coleman, B. J. Technical note: A simple model for optimizing the single machine early/tardy problem with sequence-dependent set-ups, Production and Operations Management, Vol. 1, No. 2, (1992), pp. 225-228.

[5] Allahverdi, A., Gupta, J. and Aldowaisan, T. A review of scheduling research involving set-up considerations. Omega International Journal of Management Science,Vol. 27, No, 2, (1999), pp. 219-239.

[6] M. Zandieh, S.M.T. Fatemi Ghomi, S.M. Moattar Husseini, An immune algorithm approach to hybrid flow shops scheduling with sequence dependent set-up times, Journal of Applied Mathematics and Computation V0l. 180, No. 1, (2006) pp. 111-127.

[7] Allahverdi,A., Ng, C.T., Cheng, T.C.E. , Kovalyov, M.K . A survey of scheduling problems with set-up times or costs, European Journal of Operational Research, Vol. 187, No. 3, (2008), pp. 985-1032.

[8] Choi, L.C. Korkmaz, O. Job shop scheduling with separable sequence-dependent set-ups, Annals of Operations research Vol. 70, (1997), pp. 155-170.

[9] Schutten, M. J. Practical job shop scheduling, Annals of Operations Research, Vol. 83, (1998), pp. 161-178.

[10] Naderi, B., Fatemi Ghomi, S.M.T., Aminnayeri, M. A high performing metaheuristic for job shop scheduling with sequence-dependent set-up times, Applied Soft Computing ,Vol. 10, No. 3, (2010), pp. 703-710.

[11] Ahmadizar, F, and Shahmaleki, p. Group-shop scheduling with sequence-dependent set-up and transportation times, Applied Mathematical Modelling, Vol, 38, No, 21, (2014), pp. 5080-5091.

[12] Naderi, B., Zandieh, M., Shirazi, M.A.H.A. Modeling and scheduling a case of flexible flow shops: total weighted tardiness minimization, Comput. Ind. Eng, Vol. 57, No. 4, (2009), pp. 1258-1267.

[13] Hurink, J., Knust,S. Tabu search algorithms for job-shop problems with a single transport robot, European Journal of Operational Research , Vol. 162 , No.1, (2005), pp. 99-111.

[14] Nouri, H. E., Driss, O. B., Ghédira, K. Hybrid metaheuristics for scheduling of machines and transport robots in job shop environment. Appl. Intell. Vol. 45, No. 3, (2016), pp. 808-828.

[15] Moslehi G, Mashkani O. Minimizing the Number of Tardy Jobs in the Single Machine Scheduling Problem under Bimodal Flexible and Periodic Availability Constraints. IJIEPR. Vol. 29, No .1, (2018), pp. 15-34.

[16] Schmidt, G. Scheduling with limited machine availability. European Journal of Operational Researches, Vol. 121, No. 1, (2000), pp. 1-15.

[17] Ma, Y., Chu, C., and Zuo,C. A survey of scheduling with deterministic machine availability constraints. Computers & Industrial Engineering, Vol. 58, No. 2, (2010), pp. 199-211.

[18] Vahedi-Nouri, B., Fattahi,P., Tavakkoli-Moghaddam , R., Ramezanian, R. A general flow shop scheduling problem with consideration of position-based learning effect and multiple availability constraints, International Journal of Advanced Manufacturing Technology, Vol. 73, No. 5, (2014), pp. 601-611.

[19] Eren, T., Guner, E. A bi-criteria flowshop scheduling with a learning effect. Applied Mathematical Modelling, VOL. 32, No. 9, (2008), pp. 1719-1733.

[20] Biskup,D. Single-machine scheduling with learning considerations, European Journal of Operational Research, Vol. 115, No. 1, (1999), pp. 173-178.

[21] Biskup.D. A state-of-the-art review on scheduling with learning effect, European Journal of Operational Research, Vol. 188, No. 2, (2008), pp.188 - 315.

[22] Amirian, H., Sahraeian, R. Multi-objective differential evolution for the flow shop scheduling problem with a modified learning effect. International Journal of Engineering ,Vol. 27, No. 9, (2014), pp. 1395-1404.

[23] Cheng,T., Wu, C., Chen, J., Wu, W., Cheng, S. Two-machine flow shop scheduling with a truncated learning function to minimize the make-span. International Journal of Production Economics, Vol. 141, No. 1, (2012), pp. 79-86.

[24] Salehi Mir M S, Rezaeian J. Two meta-heuristic algorithms for parallel machines scheduling problem with past-sequence-dependent setup times and effects of deterioration and learning. IJIEPR. Vol. 27, No. 1, (2016), pp.69-88.

[25] Okolowski, D., Gawiejnowicz, S. (2010). Exact and heuristic algorithms for parallel-machine scheduling with DeJong's learning effect. Computers & Industrial Engineering, Vol. 59, No. 2, (2010), pp. 272-279.

[26] Fattahi P, Keneshloo S, Daneshamooz F, Ahmadi S. A Hybrid Genetic Algorithm and Parallel Variable Neighborhood Search for Jobshop Scheduling With an Assembly Stage. IJIEPR. Vol. 30, No.1, (2019), pp. 25-37.

[27] Mirjalili, S., Mirjalili,S.M., Lewis, A. Grey wolf optimizer. Advances in Engineering Software, Vol. 69, (2014), pp. 46-61.

[28] Mehrabian, A. R., Lucas, C. A novel Numerical Optimization Algorithm Inspired from Weed Colonization, Ecological Informatics, Vol. 1, (2006), pp. 355- 366.