Uncertain Supply Chain Management 7 (2019) 599–618

Contents lists available at GrowingScience

Uncertain Supply Chain Management

homepage: www.GrowingScience.com/uscm

Meta-heuristics algorithm for two-machine no-wait flow-shop scheduling problem with the effects of learning

Faramarz Nouri[a], Saeede Samadzad[a] and Javid Ghahremani Nahr[a*]

[a] *Faculty member of Development and Planning Institute of ACECR, Iran*

C H R O N I C L E

Article history:
Received April 6, 2019
Received in revised format May 10, 2019
Accepted May 18 2019
Available online
May 18 2019

Keywords:
Uninterrupted flow-shop scheduling
Learning effect
Metaheuristic algorithms
Intermittent tasks

A B S T R A C T

In today's world, due to rapid changes in the market, scheduling as one of the most fundamental issues of competitive production, plays a very important role in maintaining the competitive position and survival of manufacturing organizations, therefore, development of scheduling models in order to improve the timing criteria is of great importance. In this research, we put the development of a no-wait flow-shop scheduling model alongside with the effect of learning into consideration to minimize the cost of consumption of resources. Finding the correct sequence of two machines' performance and optimized allocation of the resources for any performance on each machine, were considered as the main goal in this study. To solve the problem, metaheuristic genetic algorithms, particle swarm optimization, imperialist competitive algorithms, optimization of the whale and the League Champions algorithms, have been used. The statistical comparisons and also using of TOPSIS Multi-Criteria Decision Making method, indicate high level of efficiency of the League Champions algorithm with the utility weight of 0.9516.

1. Introduction

Determining the program of priority and operation sequence in production planning issues, is one of the most important key factors in each production department, since planning and scheduling of production, prevents capital accumulation, reduces the waste, decrease or eliminates idle time of machinery, and cause trying to make better use of them, optimizing energy consumption, timely responsiveness to customer orders, and supplying raw materials and components at the right time. Problems of scheduling largely varying and the purpose of production scheduling, is to allocate limited resources to perform a set of activities over the time. Having an appropriate production scheduling program, cause a decisive impact on increasing the productivity of the organization. The production scheduling model in any manufacturing organization varies according to the goals and priorities of access to each of them; as a consequence, to determine the appropriate scheduling model in the organization, the goals, priorities and constraints on the resources must first be investigated. Operation sequence and scheduling, are kind of decision-making processes have a key role to improve productivity in manufacturing and service industries. In general, scheduling refers to the process of

* Corresponding author
E-mail address: javid.ghahremani@yahoo.com (J. Ghahremani Nahr)

doi: 10.5267/j.uscm.2019.5.002

allocating a limited number of resources to carry out a limited set of activities over the time, aimed to optimize one or more performance criteria. From another point of view, it can be said that scheduling is somehow dependent on decision-making and also is a process through which scheduling determined and ultimately optimizes one or more goals and performance criteria. In most manufacturing systems or information processing environments, scheduling acts as an important decision-making process (Baker, 1974). Operation sequence is to determine the order of operation processing, and scheduling is to decide on start and end time of the operation for available resources. In today's competitive world, having the best operation sequence and appropriate scheduling of activities, is a vital requirement for companies to survive. According to Dempster (2012), scheduling defined as: "The art of allocating resources to activities, ensuring that activities carried out in a reasonable time". In practice, scheduling is made using scheduling algorithms or knowledge-based rules.

In recent years, many researchers have been attracted to no-wait issues, and this interest mostly stems from its application in the industry. A no-wait scheduling problem could occur in the production environment, when all processing levels of a task from beginning to the end must be done on the machines with no-wait and the processing steps carried out one after another without any delay. In other words, the difference between the end and start time of any task in a no-wait production environment, is equal to the total process time. Most industries such as: chemical and petrochemical, steel, glassware and paper-related industries, encounter limitations in the production process (Aldowaisan & Allahverdi, 2003). When the processing time of a task starts, the next steps have to be carried out with no delay from one machine to another. Even in the case to start work from previous step, its delay should go until the next processes start without any delay. This type of problems are so-called "no-wait flow shop" or "zero expectation flow shop". One of the two main reasons of such problems (no-wait) to occur in the production environments, refers to the nature of the processes and technology employed (Choi et al., 2008). In some processes, in order to prevent undesirable changes in temperature or other properties of materials (e.g. adhesion), it needs to do things in a continuous and on-wait manner, otherwise will not get the desired results. Practicing this issue in the service industry would only be justified, when cost of waiting time for the customers and service provider is too much. Lack of storage between machineries or workstations is the second reason for no-wait environments to appear (Chen et al., 2008). Apart from the mentioned examples above, we also encounter the problem of just in time production systems or pull systems. In other words, when a flow of tasks performed sequentially with no intermediate accumulation, there is a no-wait flow shop model in the system (Shao et al., 2017). The problem of no-wait flow shop has been studied in recent decades. Various applications of this issue in the industry, caused a lot of interest in researchers for modeling and providing solving methods. For example, several articles dating back to the 70s, reviewed the computational complexity of such issues. In recent years, most research on heuristics and meta-heuristics algorithms have been proposed to solve this type of problems.

2. Review of Literature

Hall & Sriskandarajah (1996) conducted a comprehensive overview on no-wait scheduling issues and presented some suggestions for future research. Allahverdi & Aldowaisan (2004) were the first researchers, provided both hybrid simulated annealing algorithm and hybrid genetic algorithm, aimimg at minimizing the maximum makespan and maximum delay for the no-wait flow shop problem. Tavakkoli-Moghaddam et al. (2007) presented a hybrid artificial immune system algorithm to find Pareto solutions for the no-wait flow shop problem with the objective functions of the total average weight of the completion time of the tasks and the mean of delays total weight. Eren et al. (2008) discussing the flow shop scheduling problem along with the learning effect, and also proved that their model results in the optimal answer. Asadzadeh and Zamanifar (2010) suggested compound solution approaches to solve no-wait flow shop problems and used genetic algorithms as a solution for the basic flow shop scheduling problem. In a similar approach, Chen et al. (2012) used the Genetic Algorithm to solve the sub-problem of allocating tasks to machines. There is a very limited research in the literature

of scheduling with intermittent tasks, and papers mostly focused on single-machine and parallel machines issues. Goldbogen et al. (2013) proposed a complex integer programming model for a two-objective single-machine scheduling problem. They used the branch and bound method to solve large and medium size problems. Vahedi-Nouri. (2013) developed the flowshop scheduling problem, taking into account the same processing arrangement and learning effect. They first introduced their proposed mathematic model and further developed it with its meta-heuristic algorithm. Laha and Gupta (2016) examined the no-wait flowshop scheduling problem taking into account the minimization Makespan, and initially modeled the problem with mathematical programming, and then improved it with the Hungarian algorithm. He (2016) discussed the scheduling flowshop problem with the same processing order, along with the learning effect to minimize the delay of the system, and improved it with the branch and bound algorithm. Ye et al. (2016) examined no-wait flowshop scheduling problem to maximize makespan, arguing that the problem is NP-hard in high volume, and then developed it with an efficient algorithm. Ye et al. (2017) examined no-wait flowshop scheduling system problem to minimize the completion time of the entire job, and developed the model using meta-heuristic algorithm. Ultimately, they described the solutions in both large and small scales. Chen et al. (2017) modeled the two-stage assembly flowshop scheduling problem with three machines to minimize flowshop along with learning effect, and also used ant colony algorithm to optimize in large dimensions. Shao et al. (2017) studied the no-wait flowshop problem and described the mathematical model. They used the greedy search algorithm to solve the problem. Engin et al. (2018) examined the no-wait flowshop problem and examined its mathematical model and improved it using ant meta-heuristic algorithm. Li et al. (2018) examined the no-wait flowshop problem with regard to the time-dependent preparation time and degradation effects. They used the local search algorithm to solve their proposed model. Shahvari and Logendran (2018), in their research, addressed the flowshop scheduling as a package with learning effect. In this study, first, they defined the mathematical model, and then analyzed a two-stage hybrid algorithm to minimize the total sum of completion times and total delay in the system. Gao et al. (2018) investigated no-wait flowshop scheduling problem in production paths along with learning effects, and used the meta-heuristic neighborhood algorithm to solve this model. Bai et al. (2018) examined the flowshop problem, taking into consideration the preparation time of the job, along with the learning effect, and developed it using the branch and bound algorithm.

Given the review of the related literature as well as the necessity of using meta-heuristic algorithms in solving flowshop scheduling problem, a mathematical model of no-wait flowshop scheduling problem with learning effect is presented and the meta-heuristic algorithms i.e. genetics, particle swarm optimization, whale optimization, colonial competition, and the League Champions algorithm were used to solve the problem.

3. Modeling

In this part of the paper, statement of the problem and modeling of no-wait flowshop scheduling problem, along with learning effect to minimize the cost of using resources are addressed. To do this, we continue to define the sets, parameters, and decision variables for modeling the problem.

3.1. Sets

J	Job sets	$j, r = \{1,2, \dots, J\}$
I	machines set	$i = \{1,2\}$
S	Similar machine set	$s = \{1,2, \dots, S\}$

3.2. Parameters

$\bar{p}_{ji}$	Normal process time of j on machine i (processing without regard to resource allocation and learning effect)
g_{ji}	Allocated cost to the allocated resource j on machine i

a	Learning factor due to learning
$\alpha, \beta, \gamma, \theta$	Fixed weight coefficients effective in processing operations

3.3. Decision variables

p_{ji}	The actual process time of j on machine i (considering the allocation of resources and learning effect)
c_{ji}	Completion time of j on machine i
$[j]$	The position of j in the sequence of doing jobs
u_{ji}	The amount of resources allocated to complete the job j on machine i
d	The time for doing all joint jobs
L_j	The latest time to do the job j
E_j	The earliest time to do the job j
$Cmax$	Makespan value
y_{js}	It takes 1, if the job j is assigned to the same machine s and gets zero, otherwise.
x_{jrs}	It takes 1, if the job j is assigned to the same machine s in the sequence of operations r and takes zero, otherwise.
π_{jr}	It takes 1, if the job j is assigned to the operation r and gets zero, otherwise.
ω_j	Covariate variables to determine the optimal allocation of resources
υ_j	Covariate variables to determine the optimal allocation of resources

In this problem, there are J independent job $\{1,2,\dots,J\}$ to be processed continuously on two machines $i = 1,2$, each of which has similar machines $s = \{1,2,\dots,S\}$. For example, every job must first be processed on the first machine and then on the second machine, and jobs are not allowed to wait between two machines (there is no interruption). Also, all jobs are similarly available at zero time and there is no preference for doing the jobs. The nonlinear mathematical model is presented according to the set, parameters and decision variables as follows:

$$\min \quad Z = \sum_{j=1}^{J}\sum_{r=1}^{J}\sum_{s=1}^{S} \theta^{\frac{I}{I+1}} \left(I^{-\frac{I}{I+1}} + I^{\frac{1}{I+1}}\right) \lambda_{jrs} x_{jrs} \tag{1}$$

subject to

$$c_{[j]2} \geq c_{[j]2} + p_{[j]1}, \quad \forall j \tag{2}$$

$$c_{[j]1} \geq p_{[j]1}, \quad \forall j \tag{3}$$

$$p_{ji} = \left(\frac{\bar{p}_{ji} r^{a}}{u_{ji}}\right)^{I}, \quad \forall j, i \tag{4}$$

$$d = c_{[k]}, \quad k = \min\left\{max\left\{\left\lfloor\frac{(\beta-\gamma)J}{\alpha+\beta}\right\rfloor, 0\right\}, J\right\} \tag{5}$$

$$c_{[j]i} \geq c_{[r]2} + p_{[j]i} + M * \left(\sum_{s=1}^{S} x_{jrs} - 1\right), \quad \forall j, r, i \tag{6}$$

$$cmax \geq c_{j2}, \quad \forall j \tag{7}$$

$$u^{*}_{[j]i} = \begin{cases} \left(\frac{I\omega_j}{\theta g_{[j]i}}\right)^{\frac{1}{I+1}} \left(\bar{p}_{[j]i}\right)^{\frac{I}{I+1}}, & j = 1 \quad i = 1 \\ \left(\frac{I\upsilon_j}{\theta g_{[j]i}}\right)^{\frac{1}{I+1}} \left(\bar{p}_{[j]i} J^{a}\right)^{\frac{I}{I+1}}, & j = J \quad i = 2 \\ \left(\frac{I(\omega_j + \upsilon_{j-1})}{\theta g_{[j]i}}\right)^{\frac{1}{I+1}} \left(\bar{p}_{[j]i} j^{a}\right)^{\frac{I}{I+1}}, & j > 1 \quad i = 1 \\ \left(\frac{I(\omega_{j+1} + \upsilon_i)}{\theta g_{[j]i}}\right)^{\frac{1}{I+1}} \left(\bar{p}_{[j]i} j^{a}\right)^{\frac{I}{I+1}}, & j < J \quad i = 2 \end{cases} \tag{8}$$

$$\omega_j = \begin{cases} J\gamma, & j = 1 \\ \alpha(j-1) + \gamma J, & 2 \le j \le k \\ \beta(J-j+1), & j > k \end{cases} \quad (9)$$

$$\upsilon_j = \begin{cases} -\alpha, & j \le k-1 \\ \alpha(k-1) - \beta(J-k) + \gamma J, & j = k \\ \beta, & j \ge k+1 \end{cases} \quad (10)$$

$$\sum_{r=1}^{J} x_{jrs} = y_{js}, \quad \forall j, s \quad (11)$$

$$\sum_{j=1}^{J} x_{jrs} = y_{rs}, \quad \forall r, s \quad (12)$$

$$\lambda_{jrs} = \left((\eta_r)^{\frac{1}{I+1}} r^{\frac{Ia}{I+1}} \left(g_{j1}\bar{p}_{j1}\right)^{\frac{I}{I+1}} + (\psi_r)^{\frac{1}{I+1}} r^{\frac{Ia}{I+1}} \left(g_{j2}\bar{p}_{j2}\right)^{\frac{I}{I+1}} \right) y_{js}, \quad \forall j, r, s \quad (13)$$

$$\eta_r = \begin{cases} \omega_r, & r = 1 \\ \omega_r + \upsilon_{r-1}, & r \ge 2 \end{cases} \quad (14)$$

$$\psi_r = \begin{cases} \omega_{r+1} + \upsilon_r, & r < J \\ \upsilon_r, & r = J \end{cases} \quad (15)$$

$$\sum_{s=1}^{S} y_{js} = 1, \quad \forall j \quad (16)$$

$$\pi_{jr} = \sum_{s=1}^{S} x_{jrs}, \quad \forall j, r \quad (17)$$

$$E_j = \max\{0, d - c_j\}, \quad \forall j \quad (18)$$

$$L_j = \max\{0, c_j - d\}, \quad \forall j \quad (19)$$

$$x_{jrs}, y_{js}, \pi_{jr} \in \{0,1\}, \quad \forall r, j, s \quad (20)$$

$$\psi_r, \eta_r, \lambda_{jrs}, \omega_j, u_{ji}, d, E_j, L_j, p_{ji}, cmax, c_j \ge 0, \quad \forall j, r, s \quad (21)$$

$$\upsilon_j, \quad \forall j \ free \quad (22)$$

Eq. (1) shows the value of the objective function of the model and aims to minimize the cost of using resources by taking into account the learning effect in the correct sequence of operations in an uninterrupted flowshop. Constraints (2) and (3) ensure that the waiting time between the two operations is zero; there is no interruption in the other. Constraint (4) shows the actual processing time of each job on each machine according to the available resources and the learning effect. Constraint (5) ensures that there is an optimal sequence at which time all joint jobs are equal to the completion time of the job. Constraint (6) shows the completion time of the jobs after the second machine operation. Constraint (7) shows the makespan value. Constraint (8) shows the optimal allocation of resources for each job on each machine according to the learning effect. Constraints (9) and (10) show the auxiliary variables related to the optimal allocation of resources. Constraints (11) and (12) ensure that each job can only be in a sequence of operations. Constraints (13) to (15) are covariate variables for problem solving. Constraints (16) show that each job can only be assigned to a similar machine of type 1 and type 2 machines. Constraints (17) show the order of the sequence of each done task. Constraints (18) and (19) calculate the earliest and latest time for each job. Constraints (20) to (22) express the type and gender of the decision variables.

4. Problem solving method

This section outlines the foundations of the meta-heuristic algorithms used to solve the problem, including genetic algorithm, particle swarm optimization algorithm, colonial competition algorithm, whale optimization algorithm and League Champions algorithm.

4.1. Genetic Algorithm

The genetic algorithm begins by randomly generating a primitive population of chromosomes, while satisfying the limits or constraints of the problem. In other words, chromosomes are strings of the proposed values for problem-solving variables, each representing a probable answer to the problem. The chromosomes are deduced from successive replicates called generations. During each generation, these chromosomes are evaluated according to the optimization objective, and the chromosomes that are considered to be the best answer to the problem are more likely to reproduce the problem. It is very important to formulate the chromosome fitness function in order to help accelerate the convergence of computations towards the optimal public response.

In this study, for solving no-wait flowshop scheduling problem with learning effect to minimize the cost of using the total resources from the chromosome presented in Figure 1, is used. In this figure, a permutation has been created from the number of jobs, which is the priority of doing job on the job considered by the machines.

Job	1	2	3	4	5	6	7
Priority Job on Machine	2	4	5	1	3	6	7
Machine Allocadet	1	2	2	2	1	2	1

Fig. 1. Primary chromosome used to solve the no-wait flowshop scheduling problem

As shown in Fig. 1 jobs are considered for the flowshop. The way to decrypt is that the lowest stated priority is the first job done on the machines. That is, according to Fig. 1, job 4 must first be performed by machines (similar machine no. 2 from the set of machines number 1). Jobs 1-5-2-3-6-7 are the next priorities of the user on the machines. In this study, a single-point mutation and two different crossovers are used as follows. Also, due to the continuous nature of meta-heuristic algorithms and the discrete nature of the initial solution, first, using the conversion operator, the continuous response generated by the algorithms must be converted into a discrete solution of the problem. Figure 2 represents how to convert the algorithm's continuous response to a discrete solution. Suppose that the initial answer contains a total of 7 different jobs from a scheduling problem.

continuous solution	5.40	5.01	6.81	7.38	7.78	0.12	3.21
Conversion Operator	4	3	5	6	7	1	2

Fig. 2. Operator for converting a continuous solution to a discrete one

According to Fig. 1, to convert a continuous solution to a discrete solution, the smallest number of continuous responses is selected (value 0.12) and job 1 is replaced. Then, the next job (2) is assigned instead of the largest next number of the continuous solution (value 3.21). This will continue until all the jobs are numbered. Figure 3 shows the performance of the Type-1 crossover for the 7 assumed jobs.

Parent 1	6.8	2.4	2.0	5.4	9.6	4.0	6.2
Conversion Operator	6	2	1	4	7	3	5

Child 1	6.8	2.4	2.0	5.4	7.2	9.0	9.6
Conversion Operator	4	2	1	3	5	6	7

Parent 2	3.0	7.7	3.2	5.5	7.2	9.0	9.6
Conversion Operator	1	5	2	3	4	6	7

Child 2	3.0	7.7	3.2	5.5	9.6	4.0	6.2
Conversion Operator	1	6	2	4	7	3	5

Fig. 3. Type-1 crossover operator

In Type-1 crossover, a point is selected from the parent chromosome and with the constant holding of the first part of the parent chromosome, the second part of the parent chromosome is displaced by constant or variable. Thus, two new offsprings are created from the parents. Fig. 4 also shows how Type-2 crossover performs for the seven assumed jobs.

Parent 1	6.8	2.4	2.0	5.4	9.6	4.0	6.2
Conversion Operator	6	2	1	4	7	3	5

Child 1	-0.8	7.7	3.2	5.5	4.8	9.0	9.6
Conversion Operator	1	5	2	4	3	6	7

Gap	3.8	-5.3	-1.2	-0.1	2.4	-5.0	-3.4

Parent 2	3.0	7.7	3.2	5.5	7.2	9.0	9.6
Conversion Operator	1	5	2	3	4	6	7

Child 2	10.6	2.4	2	5.4	12	4	6.2
Conversion Operator	6	2	1	4	7	3	5

Fig. 4. Type-2 crossover operator

In type-2 crossover, the difference between the genes of the parent is firstly calculated and called d. Then, using the following formula, two new offsprings are created from the parent. Also α provided in the following statement can be constant or variable:

$$chile1 = \min\{(parent1, parent2) - \alpha.* d \\ chile2 = \max\{(parent1, parent2) + \alpha.* d \quad (23)$$

Finally, Fig. 5 shows how to perform a single-point mutation operator for a 7-job designed model.

Parent 1	6.8	2.4	2.0	5.4	9.6	4.0	6.2
Conversion Operator	6	2	1	4	7	3	5

Child 1	6.8	2.4	2.0	5.4	7.2	9.0	4.3
Conversion Operator	5	2	1	4	6	7	3

Parent 2	3.0	7.7	3.2	5.5	7.2	9.0	9.6
Conversion Operator	1	5	2	3	4	6	7

Child 2	3.0	7.7	3.2	5.5	9.6	4.0	1.8
Conversion Operator	2	6	3	5	7	4	1

Fig. 5. single-point mutation operator

In this type of operator, 1 gene is randomly selected from the parent chromosome and replaced by a new random number. After doing this, the conversion operator is used to convert a continuous response to a discrete solution

4.2. Imperialist Competitive Algorithm

There is a method in the field of evolutionary computing that addresses the optimal answer to various problems of optimization and was first introduced by Atashpaz-Gargari and Lucas (2007). In terms of application, this algorithm goes under the rubri of evolutionary optimization algorithms. Like all algorithms in this category, the colonial competition algorithm also makes up a set of possible solutions. These initial solutions are known as (country) in colonial competition algorithm. The colonial competition algorithm with a particular process improve the initial answers (countries) and ultimately provides the optimal answer to the optimization problem. The main pillars of this algorithm are the policy of alignment, colonial competition and revolution. This algorithm, by imitating the social, economic, and political evolution of countries, and using mathematical modeling, provide parts of this process with regular operators in the form of algorithms that can help solve complex optimization problems.

4.3. Particle swarm optimization Algorithm

Particle swarm motion algorithm divides the solution space using a quasi-likelihood function into multi-path paths, which are formed by the movement of individual particles in space. The movement of a group of particles consists of two main components (definite and probable). Each particle is interested in the direction of the best current answer x^* or the best answer so far g^*. For every particle

moving in space, regardless of whether it meets swarm intelligence, there are vectors of space and speed. Now for particle i (bird) that moves with the use of swarm intelligence, if the vector of its current location is equal to x_i, then the vector of the motion velocity, which is displayed as v_i, is, according to Eq. (24), determinable. Also, the vector of the new location of each particle is also as Eq. (25).

$$v_i^{t+1} = v_i^t + \alpha\epsilon_1 \odot \left[g^* - x_i^t\right] + \beta\epsilon_2 \odot \left[x_i^* - x_i^t\right] \quad (24)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (25)$$

In this relation ϵ_1 and ϵ_2 are random vectors, the values of their arrows are real numbers between zero and one. Also, the sign $\odot$ represents the internal multiplication between the two matrices. Parameters α and β are considered as learning parameters and acceleration parameters.

4.4. League Champions algorithm

In the League Champions algorithm, a set of L answers is randomly selected from the search space. Each answer from the population belongs to one of the teams (L is an even number), which represents the current arrangement of the team. Therefore, team i represents member i of the population. Each answer from the population has its own specific fitness. During this algorithm, different solutions that can be given to a problem are compared based on their degree of fitness (values of their objective function) and each one is improved and finally a solution near optimal is chosen. A number of teams (metaphor of the solutions under study) in the form of a league (metaphor of the population of possible solutions) compete with each other in a few weeks (the metaphor of the number of evaluation steps in a repetition of the algorithm) and compete two by two. Based on the power of the game (the metaphor of the degree of fitness or the value of the objective function of the solution vector) from the team arrangement (the metaphor of its possible code for the problem), the winning and losing teams are determined (Kashan & Karimi, 2010). Each week, each team by its coach, with the process of artificial analysis of the games of the previous week and with the help of the best arrangement until that time, reaches a new team arrangement (metaphor of creating new answers). Thus, the competition for the championship will continue for several seasons (metaphor of the number of algorithm repetitions) (Kashan, 2014). The number of seasons (S) and the number of teams (L) are adjustable parameters that have a direct effect on their final response to the algorithm.

4.5. Whale optimization algorithm (WOA)

Humpback whales can detect the location of the bait and surround them. Since the position of optimal design in search space is not known in advance, the WOA algorithm assumes that currently the candidate for the best solution is the target or is near optimal (Mirjalili & Lewis, 2016). Once the search engine is defined, other search factors will try to update their position on the best search engine. This behavior is shown by the following equations:

$$\vec{D} = |\vec{C}.\vec{X}(t) - \vec{X}(t)| \quad (26)$$

$$\vec{X}(t+1) = \overrightarrow{X^*}(t) - \vec{A}.\vec{D} \quad (27)$$

In the above relations, t shows the current repetition, $\vec{A}$ and $\vec{C}$ are the vector of coefficients, X^* is the vector of the position of the best solution so far and $\vec{X}$ is the vector of the object's position. It should be noted that X^* should be updated every time, if there is a better solution, vectors $\vec{A}$ and $\vec{C}$ are calculated as follows:

$$\vec{A} = 2\vec{a}.\vec{r} - \vec{a} \quad (28)$$

$$\vec{C} = 2\vec{r} \quad (29)$$

Where $\vec{a}$ reduces linearly from 2 to 0 during repetitions and is a random vector. $\vec{r}$ is random vector [0,1]. It is worth noting that WOA only contains two main internal parameters that must be set (A, C).

5. Computational results

This section addresses the problem and analyzes the results. First and foremost, a small scale problem has been designed and solved in GAMS software, and the output variables from the problem, including the sequence of operations, are shown. In the next step, the meta-heuristic algorithms proposed in this study are tuned by Taguchi method and then several sample problems sre designed and solved in different sizes to measure their efficiencies. Finally, using Tukey statistical test, the significance of the means of the objective function and the computational time are investigated and the TOPSIS method is used to select the most efficient algorithm.

5.1. Solving sample problem in small size

In this section, in order to examine the proposed model and validate it, as well as review the output variables, a small size sample consisting of 7 jobs, 2 machines, each machine with 3 similar machines has been designed. In this sample, learning coefficient is equal to $a = -0.322$ and effective weight coefficients in the processing of operations are equal to $\alpha = 7$, $\beta = 17$, $\gamma = 5$, $\theta = 1$. Table 1 also shows the normal process time, as well as the cost allocated to the dedicated resource to the machine.

Table 1
The data used for the small size sample

J	j_1	j_2	j_3	j_4	j_5	j_6	j_7
$\bar{p}_{j1}$	4	6	8	9	10	5	18
$\bar{p}_{j2}$	1	3	5	6	4	17	3
g_{j1}	2	5	3	1	6	7	4
g_{j2}	3	4	2	4	5	6	8

By solving the above problem, the value of the objective function of the problem is 598.860 and the makespan is 2.828. Table 2 shows the order of the sequence of the jobs on the machines, respectively.

Table 2
Output variable π_{jr} resulting from small size problem solving

pi(r,j)	r_1	r_2	r_3	r_4	r_5	r_6	r_7
j_1	1	0	0	0	0	0	0
j_2	0	0	1	0	0	0	0
j_3	0	1	0	0	0	0	0
j_4	0	0	0	0	1	0	0
j_5	0	0	0	1	0	0	0
j_6	0	0	0	0	0	0	1
j_7	0	0	0	0	0	1	0

The output of Table 2 shows that the order of the sequence of operations on two machines is as follows: 1-6-3-4.5-4.7-6. Table 3 also shows the earliest and the latest operation time and the amount of resources allocated to each machine.

Table 3
The output variable E_j,L_j, and u_{ji} derived from solving the small size problem

	j_1	j_2	j_3	j_4	j_5	j_6	j_7
E_j	0.527	0.281	0.299	0	0	0	0
L_j	0	0	0	0	0.024	2.179	1.179
i_1	8.243	5.858	9.595	14.814	8.716	4.862	11.630
i_1	2.858	4.945	8.452	7.446	4.874	10.109	2.219

5.2. Sensitivity analysis of the problem

In order to analyze the sensitivity of the problem and to observe changes in the Makespan value and the value of the objective function, all of the coefficients of learning effect and effective weighing in the process of its measured operation are discussed. So in this section, 5 different sensitivity analyses are performed as follows:

5.2.1. Sensitivity analysis of the problem on a coefficient

At first, the sensitivity analysis of the problem has been addressed on a coefficient. In 11 different scenarios, the changes in the value of the objective function and the magnitude of makespan have been analyzed. Table 4 shows the magnitude of the changes in the objective function and the amount of makespan in 11 different scenarios.

Table 4
Sensitivity analysis of the problem on a coefficient

Scenario	a coefficient	Objective Value	$Cmax$
1	-0.161	694.30	1.925
2	-0.193	674.47	2.078
3	-0.225	655.18	2.243
4	-0.257	636.15	2.241
5	-0.290	616.90	2.620
6 (Base)	-0.322	589.85	2.828
7	-0.354	581.31	3.053
8	-0.386	564.16	3.295
9	-0.418	547.59	3.557
10	-0.450	531.58	3.840
11	-0.483	515.64	4.155

Given the results of Table 5, it can be seen that with the increase of the learning effect coefficient, the value of the objective function is reduced, whereas the magnitude of the makespan is increased. Figure 6 shows the trend of changes in the value of the objective function and the magnitude of the Makespan in different scenarios.

Fig. 6. Trend of changes in the value of the objective function and c_{max} due to changes in $\boldsymbol{a}$ coefficient

5.2.2. Sensitivity analysis of the problem on coefficients $\alpha, \beta, \gamma, \theta$

In the following, the sensitivity analysis of the problem is considered on the coefficients $\alpha, \beta, \gamma, \theta$ and in seven different scenarios, the changes in the value of the objective function and the magnitude of the Makespan are analyzed. Table 5 gives the amount of changes in the objective function and makespan value in seven different scenarios.

Table 5
Sensitivity analysis of the problem on coefficients $\alpha, \beta, \gamma, \theta$

Scenario		1	2	3	4 (Base)	5	6	7
α	Value	4	5	6	7	8	9	10
	Objective Value	587.77	591.62	595.42	598.85	601.98	605.02	607.91
	$Cmax$	2.828	2.828	2.828	2.828	2.828	2.828	2.828
β	Value	11	13	15	17	19	21	23
	Objective Value	553.51	573.23	587.76	598.85	609.10	618.52	623.40
	$Cmax$	3.780	3.382	3.074	2.828	2.626	2.456	2.312
γ	Value	2	3	4	5	6	7	8
	Objective Value	519.76	555.42	580.42	598.85	615.35	629.11	638.65
	$Cmax$	2.828	2.828	2.828	2.828	2.828	2.828	2.828
θ	Value	0.4	0.6	0.8	1	1.2	1.4	1.6
	Objective Value	325.11	426.01	516.08	598.87	676.25	749.45	819.22
	$Cmax$	1.535	2.012	2.437	2.828	3.194	3.539	3.869

In accordance with the results of Table 5, with the increase of coefficient α and θ, the value of the objective function increases, and with the increase of the coefficient β ، the value of the objective function is reduced. Moreover, the magnitude of Makespan is increased by increasing the learning effect and coefficient θ and increases by increasing β, remains unchanged with decreasing or increasing coefficients γ and α. Fig. 7 shows the trend of changes in the variations of the amount of the objective function and the amount of makespan in different scenarios for the coefficients $\alpha, \beta, \gamma, \theta$.

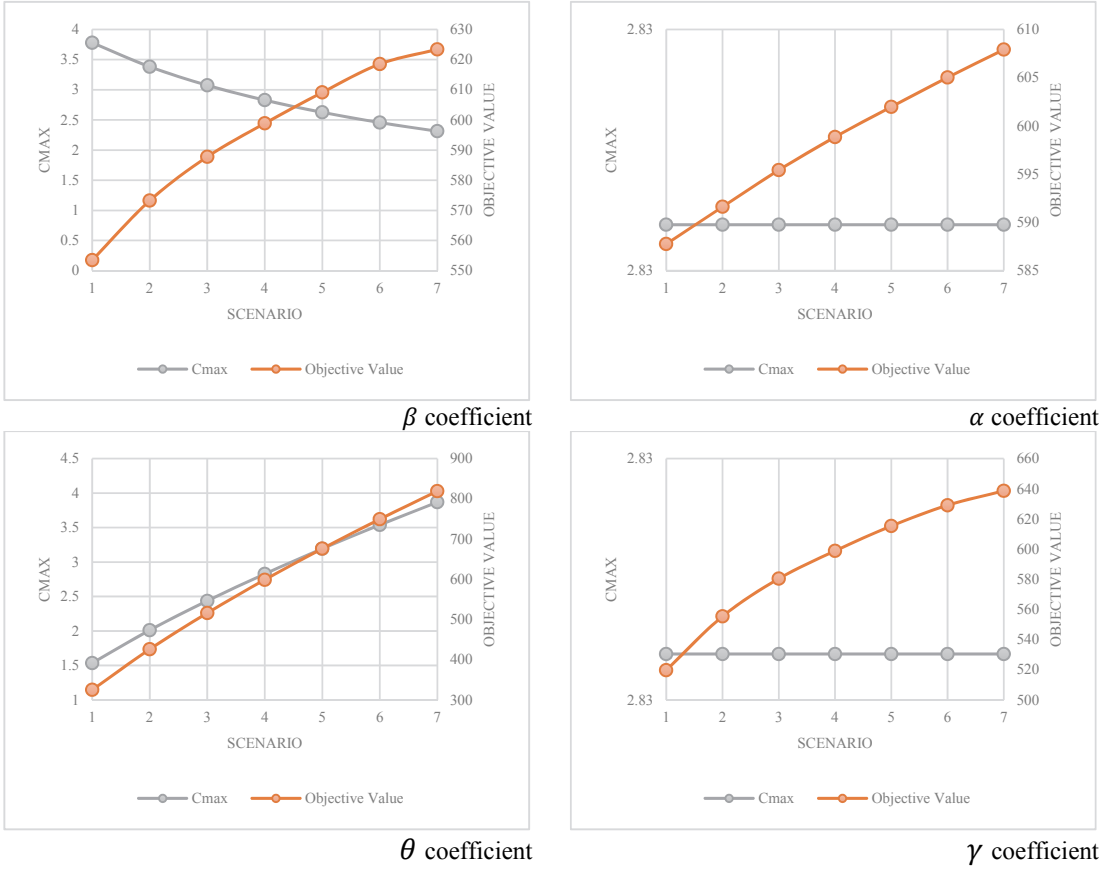

Fig. 7. Trend of changes in the value of the objective function and C_{max} due to changes in coefficients $\alpha, \beta, \gamma, \theta$

5.3. Sample problem solving with meta-heuristic algorithms

In this section, in order to solve problems using meta-heuristic algorithms, the sample small size problem has been analyzed and the output variables derived from the meta-heuristic algorithms and the GAMS software are compared with each other.

5.3.1. Parameter tuning of meta-heuristic algorithms by Taguchi method

Before solving sample problems in different sizes with meta-heuristic algorithms, first of all, the initial parameters of the meta-heuristic algorithms must be optimized in their optimal state. The parameterization of the meta-heuristic algorithms in order to increase the efficiency of the algorithms is to find the optimal solution in the justified space of the problem. First, the factors of the meta-heuristic algorithms (initial parameters) and the levels of its factors (current suggestions) are identified. Then using the Taguchi method and MINITAB software, the obtained analysis data are analyzed and the two ratio diagrams S/N and mean means are obtained. Given the minimization of the designed model, if each of the levels of factors considered in the mean ratio of S/N is at the highest point; the value of the order is identified as the optimal parameter of the algorithm. Table 6 shows the operating levels used, as well as the optimal value of each parameter of the genetic algorithm, particle swarm optimization, colonial competition, whale optimization and League Champions algorithm.

Table 6
Operating levels used for proposed algorithms

Algorithm		Operating levels			Optimum
	Parameters	1	2	3	
WOA	A	**0.2**	0.2	0.7	0.2
	C	0.2	0.5	**0.7**	0.7
	$Nwhale$	100	150	**200**	200
	$Max\ it$	100	150	**200**	200
LCA	$LeagueSize$	100	150	**200**	200
	$Max\ it$	100	150	**200**	200
	ψ_1	2	1	**6**	6
	ψ_2	0.9	**1**	1.1	1
	p_c	0.5	0.7	**0.9**	0.9
PSO	C_1	**1**	1.5	2	1
	C_2	**1**	1.5	2	1
	w	0.8	0.9	**1**	1
	$Nparticle$	100	150	**200**	200
	$Max\ it$	100	150	**200**	200
GA	$Npop$	100	150	**200**	200
	$Max\ it$	**100**	150	200	100
	p_c	0.3	0.5	**0.7**	0.7
	p_m	0.3	**0.5**	0.7	0.5
ICA	$Ncoun$	100	150	**200**	200
	$Nimp$	100	150	**200**	200
	$RevRate$	0.3	0.5	**0.7**	0.7
	$DefRate$	0.3	0.5	**0.7**	0.7

Minitab 16 software was used to analyze the results using Taguchi method. How to arrange the experiments can be obtained referring to this software. Each of the experiments has been repeated five times. Before transferring results to the MINITAB software, it is best to first normalize the results. In this research, normalization is done using RPD method and the following formula:

$$RPD = \frac{|\bar{y}_i - \min(\bar{y}_i)| * 100}{\min(\bar{y}_i)} \tag{30}$$

Since the objective function is minimized, the highest value of criterion S/N is the criterion for selecting the values of the parameters. According to Table 7, by comparing the difference between the maximum and minimum values obtained for the index S/N, the significant effect of the parameters of each algorithm, as well as the effect of each parameter of the algorithm, are shown in obtaining optimal results.

Table 7
Analysis of the results of parameter tuning and the order of the effect of each parameter of the algorithm

Algorithm	Parameters	Level 1	Level 2	Level 3	Δ
WOA	A	0.2305	0.3663	0.3980	0.1358
	C	0.3567	0.2696	0.2694	0.0880
	$Nwhale$	0.4615	0.3647	0.0685	0.3930
	$Max\ it$	0.4252	0.3663	0.1032	0.3219
LCA	$LeagueSize$	0.3689	0.3558	0.3121	0.0568
	$Max\ it$	0.4273	0.4065	0.2030	0.2243
	ψ_1	0.4207	0.2517	0.3643	0.1690
	ψ_2	0.3814	0.3125	0.3429	0.0688
	p_c	0.4298	0.3364	0.2706	0.1592
PSO	C_1	0.2681	0.4276	0.3730	0.1595
	C_2	0.2278	0.3540	0.4871	0.2593
	w	0.3767	0.4220	0.2702	0.1518
	$Nparticle$	0.5128	0.3843	0.1717	0.3412
	$Max\ it$	0.3966	0.3785	0.2963	0.1030
GA	$Npop$	0.2668	0.1880	0.0430	0.2264
	$Max\ it$	0.1720	0.2141	0.1091	0.1051
	p_c	0.1850	0.1960	0.1141	0.0819
	p_m	0.1497	0.2027	0.1427	0.0560
ICA	$Ncoun$	0.3334	0.2603	0.0770	0.2564
	$Nimp$	0.2720	0.2541	0.1446	0.1273
	$RevRate$	0.2273	0.2327	0.2108	0.0219
	$DefRate$	0.2163	0.2350	0.2194	0.0186

5.3.2. Solving sample problems in smaller sizes by solving methods

In order to better evaluate the results obtained from the precise method and the meta-heuristic algorithms, 8 other sample problems are designed in small sizes according to Table 8. The minimum value of the objective function and the computational time obtained in 5 consecutive repetitions of the precise method (GAMS software) and the meta-heuristic algorithms are shown in Table 9.

Table 8
Small size problems designed

Sample Problem	$(I \times J \times S)$	Sample Problem	$(I \times J \times S)$
1	8×2×3	5	15×2×5
2	9×2×3	6	18×2×6
3	10×2×4	7	20×2×7
4	12×2×4	8	22×2×8

Table 9
Value of objective function and computational time obtained from GAMS in solving sample problems

Sample Problem	Objective Value	Cpu_time	Sample Problem	Objective Value	Cpu_time
1	631.58	1.203	5	1312.8	4.23
2	710.1	1.547	6	1548.83	9.025
3	781.51	2.205	7	1853.32	41.52
4	977.69	2.677	8	2132.37	>1000

According to the results of Table 9, it is observed that the computational time of the problem solving with the GAMS software is exponentially expanded, and the GAMS software cannot find the optimal response in less than 1000 seconds in solving problems larger than the sample problem no. 8. Hence, this problem is recognized as the biggest problem in the GAMS software that has been able to solve.

Table 10
Value of objective function and computational time obtained from solving methods in solving sample problems

Solution method	Index	1	2	3	4	5	6	7	8
GAMS	Obj Value	631.58	710.1	781.51	977.69	1312.8	1548.83	1853.32	2132.73
	Cpu_time	1.2	1.54	2.2	2.67	4.23	9.02	41.25	>1000
WOA	Obj Value	631.58	710.1	781.51	977.69	1312.8	1549.64	1853.67	2134.94
	Cpu_time	6.07	6.80	7.09	9.69	12.31	15.51	17.41	21.23
LCA	Obj Value	631.58	710.1	781.51	977.69	1314.15	1550.93	1856.67	2136.14
	Cpu_time	5.09	5.45	6.09	7.39	9.62	12.14	14.18	16.20
PSO	Obj Value	631.58	710.1	781.51	977.69	1313.01	1550.41	1855.99	2134.28
	Cpu_time	5.36	6.13	6.55	7.57	10.45	12.99	15.14	17.02
GA	Obj Value	631.58	710.1	781.51	977.69	1312.94	1551.48	1855.02	2135.89
	Cpu_time	6.69	7.34	8.10	9.79	12.54	16.15	18.93	21.60
ICA	Obj Value	631.58	710.1	781.51	977.69	1312.81	1550.81	1855.26	2135.36
	Cpu_time	8.72	9.20	10.17	11.89	15.10	19.57	22.01	25.06

To illustrate the difference between the results of the meta-heuristic algorithms from the point of view of the indicators of the objective function and the computational time, Fig. 8 and Fig. 9 have been plotted. These diagrams show the difference between the minimum value of the objective function obtained from 5 replications of GAMS software and meta-heuristic algorithms and the computational time variations obtained in sample problems 1 through 8.

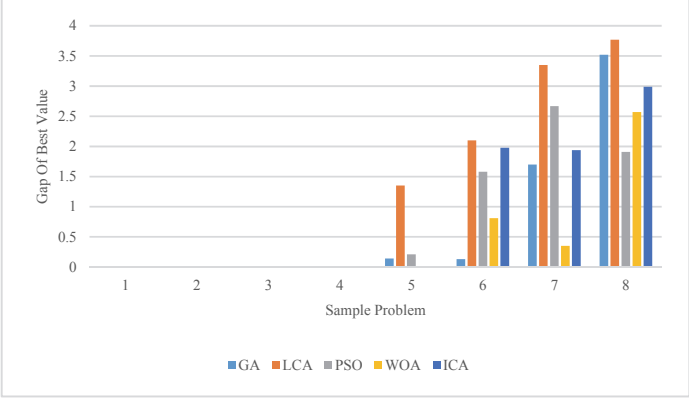

Fig. 8. The difference between the minimum values of the objective function obtained from meta-heuristic algorithms in GAMS software

With regard to Fig. 8, it is observed that meta-heuristic algorithms can reach the optimal value of the objective function for sample problems. By increasing the size of the problem and its complexity, the difference between the optimal value of the objective function less than 4 units is reported.

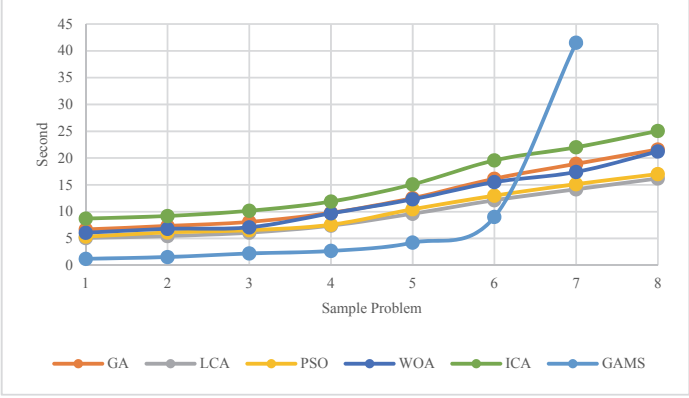

Fig. 9. Computational time obtained from solving sample problems by solving methods

As shown in Fig. 9, GAMS software has the least amount of computational time in solving typical problems up to sample problem 6. By increasing the size of the problem, from the problem sample 7, it has an exponential trend so that it has solved sample problem 8 in more than 1000 seconds. Figure 10 also shows the trend of changes of meta-heuristic algorithms in obtaining optimal results for the sample problem in 200 consecutive repetitions for the last small sample size problem.

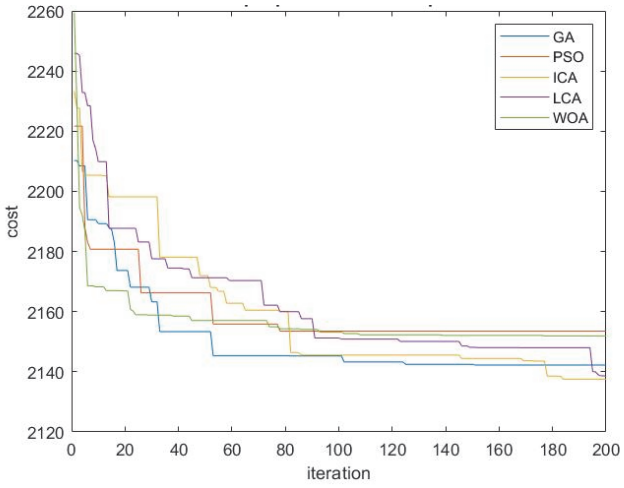

Fig. 10. The Trend of changes in the meta-heuristic algorithms in obtaining optimal solution in 200 consecutive repetitions

Tukey multiple test was run to examine the mean of the obtained solutions of the problems using the GAMS software and the meta-heuristic algorithms. For this purpose, the mean of five replications obtained from solving sample problems in small size as a basis for statistical computation has been investigated. Tables 11 and 12 show, respectively, the results of Tukey multiple statistical tests between solving methods for objective function indicators and computational time, respectively at a confidence level of 95%.

Table 11
Summary of Tukey statistical test to examine the meaningfulness of the objective function means between solving methods

Solution methods	Means Different	L_bound	U_bound	T-value	P_value
GAMS-WOA	2	-574	569	0.01	0.994
GAMS-LCA	2	-573	569	0.01	0.994
GAMS-PSO	3	-574	569	0.01	0.993
GAMS-GA	2	-573	569	0.01	0.995
GAMS-ICA	2	-573	569	0.01	0.994
WOA-LCA	0	-573	573	0.00	1.000
WOA-PSO	0	-574	573	0.00	0.999
WOA-GA	0	-572	573	0.00	0.999
WOA-ICA	0	-573	573	0.00	1.000
LCA-PSO	0	-574	573	0.00	0.999
LCA-ICA	0	-573	573	0.00	1.000
LCA-GA	0	-572	573	0.00	0.999
PSO-GA	1	-572	574	0.00	0.998
PSO-ICA	0	-573	574	0.00	0.999
ICA-GA	0	-572	574	0.00	0.999

According to the results of the above table and P-value, it is observed that there is no significant difference between any of the methods of solving means due to the higher value of P-value of 0.05. Therefore, it can be concluded that the effectiveness of the meta-heuristic algorithms in solving sample problems is very high and can be used to solve larger sample size problems. Table 12, as in Table 11, examines the significant difference between the computational time means of the solving methods.

Table 12
Summary of Tukey statistical test to examine the significance of computing time intervals between solving methods

Solution methods	Means Different	L_bound	U_bound	T-value	P_value
GAMS-WOA	1161	725	1597	6.14	0.000
GAMS-LCA	1163	727	1599	6.15	0.000
GAMS-PSO	1162	726	1598	6.15	0.000
GAMS-GA	1160	724	1596	6.14	0.000
GAMS-ICA	1158	722	1593	6.12	0.000
WOA-LCA	2.27	-2.83	7.37	0.96	0.355
WOA-PSO	1.73	-3.47	6.92	0.71	0.489
WOA-GA	0.67	-6.41	5.07	0.25	0.807
WOA-ICA	3.14	-9.22	2.94	1.10	0.289
LCA-PSO	0.55	-5.00	3.90	0.26	0.797
LCA-ICA	5.41	-10.95	0.12	2.11	0.055
LCA-GA	2.94	-8.05	2.17	1.24	0.237
PSO-GA	2.40	-7.60	2.81	0.98	0.342
PSO-ICA	4.87	-10.48	0.75	1.86	0.084
ICA-GA	2.47	-8.56	3.62	0.86	0.401

Furthermore, the results of multiple statistical tests showed that there is a significant difference between the computational times. There is a significant difference between all the computational time intervals of the GAMS software and the meta-heuristic algorithms. Since all the computational time means of the meta-heuristic algorithms are less than that of GAMS, it can be concluded that the high efficiency of meta-heuristic algorithms is proved in less time to obtain optimal results. Finally, Table 13 shows the relative difference between the values of the objective function obtained from the meta-heuristic algorithms in each repetition with those of GAMS software.

Table 13
Error percentage of relative difference between meta-heuristic algorithms using GAMS Software

Sample Problem	GA	LCA	PSO	WOA	ICA
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0.011	0.012	0
4	0.008	0.0008	0.014	0.011	0.006
5	0.134	0.151	0.111	0.157	0.079
6	0.164	0.248	0.212	0.169	0.221
7	0.293	0.358	0.291	0.307	0.268
8	0.323	0.289	0.592	0.429	0.435

5.4. Solving sample problems in larger sizes with meta-heuristic algorithms

In this section, in order to select the most efficient algorithm from whale optimization algorithms, the Champions League, Particle Swarm Optimization, Genetic and Colonial Competition, 10 sample problems are designed as shown in Table 14 and mean value of the objective function obtained together with the mean computational time are shown as in Tables 15 and 16.

Table 14
The size of designed problems in a larger size

Sample Problem	$(I \times J \times S)$	Sample Problem	$(I \times J \times S)$
1	30×2×10	6	55×2×20
2	35×2×12	7	60×2×22
3	40×2×14	8	65×2×24
4	45×2×16	9	70×2×26
5	50×2×18	10	80×2×30

Table 15
Mean value of the objective function obtained from the algorithm in solving large-size problems

Sample Problem	WOA	LCA	PSO	GA	ICA
1	2910.59	2914.37	2938.71	2901.44	2891.23
2	3560.85	3574.78	3577.86	3557.09	3576.39
3	4000.81	4012.45	4017.48	3994.18	4014.63
4	4579.35	4639.41	4653.19	4585.22	4657.44
5	5237.84	5266.25	5468.89	5223.58	5307.87
6	5852.95	5955.53	5925.72	5872.45	5984.36
7	6421.69	6445.05	6434.48	6397.89	6478.25
8	7080.71	7131.29	7189.72	7068.98	7208.16
9	7870.47	7856.18	7997.25	7909.17	7947.20
10	9006.76	9045.85	9163.28	8952.57	9043.95

Table 16
Mean value of the objective function obtained from the algorithm in solving large-size problems

Sample Problem	WOA	LCA	PSO	GA	ICA
1	34.30	26.73	29.30	36.16	39.23
2	49.51	35.41	38.59	47.07	50.88
3	58.40	43.92	48.34	59.16	62.68
4	71.52	53.79	58.23	71.34	76.17
5	84.64	64.29	69.60	85.56	90.44
6	102.90	76.91	79.34	102.92	107.05
7	120.23	90.48	93.51	120.44	124.10
8	141.29	106.41	109.29	138.02	144.11
9	156.25	118.03	125.30	157.26	160.25
10	204.30	155.11	161.30	200.54	204.23

To better compare the results above, the diagrams in Figures 11 and 12 show, respectively, the means of the objective function and the computational time of the meta-heuristic algorithms in solving large sample size problems.

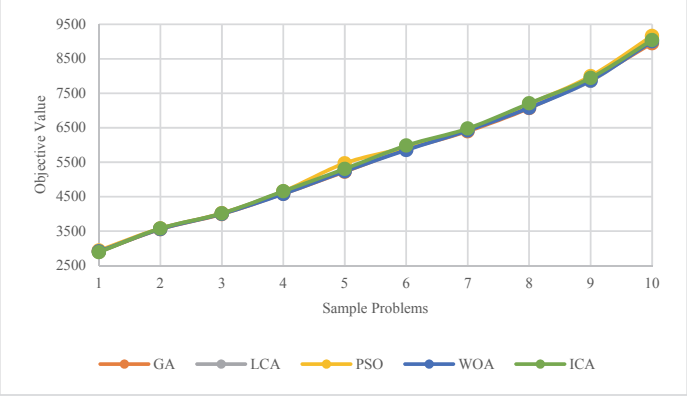

Fig. 11. Means of the objective function in solving large sample problems

Fig. 12. Computing time means in solving large sample problems

According to the results of the tables and diagrams in Figs. 11 and 12, it can be concluded that in general genetic algorithms perform better than the rest in achieving the optimal results of the objective function and that League Champions algorithm outperformed others in solving problems at a shorter time. Nevertheless, in order to select the most efficient algorithm, TOPSIS multi-criteria decision-making method and Shannon entropy has been used. For this purpose, the indexes used include the mean of the entire objective function and the computational time in large sample problems. Table 17 shows the mean of the objective function and the computational time for large sample sizes.

Table 17
Total means of objective function and computational time indices in large sample problems

Index	WOA	LCA	PSO	GA	ICA
Means of Objective value	5652.20	5684.11	5736.65	5646.25	5710.94
Means of Cpu_time	102.33	77.10	81.28	101.84	105.91
utility weight	0.1613	0.9516	0.8187	0.1784	0.0333

The results from TOPSIS showed that the League Champions algorithm has been selected as the most efficient algorithm with the utility weight of 0.9516, considering the two indicators.

6. Conclusion

In this study, a no-wait flowshop scheduling problem along with learning effect was modeled and solved to minimize the costs of the use of resources. The primary goal was to find a sequence of jobs in order to minimize makespan, as well as to find the optimal resources assigned to each job by machines. After modeling, to solve the problem, GAMS software was used and a small sample problem was analyzed and examined. Then, sensitivity analysis on the model was performed by changing the five main parameters of the model, which resulted in an increase in the value of the objective function by increasing the coefficient α، γ and θ as well as reducing the value of the objective function by increasing the coefficient of learning effect and coefficient β. Also, the magnitude of Makespan is also increased by increasing the learning effect and coefficient $\theta$and decreased by increasing coefficient β, and is unchanged with increased or decreased coefficients α and γ. The next step was to use meta-heuristic algorithms to solve problems in larger sizes. For this purpose, the parameters of the meta-heuristic algorithms were firstly tuned using Taguchi method, and eight sample problems were designed in small sizes. GAMS software solves the sample problem no.8 more than 1000 seconds. However, the results of the statistical test showed a significant difference between the GAMS software computing time intervals and the meta-heuristic algorithms. At this stage of the study, the efficiency of the meta-heuristic algorithms was confirmed to solve larger sample size problems. In the last step, 10

sample problems were designed in a larger size, which were solved using meta-heuristic algorithms. The TOPSIS method was also used to select the most efficient algorithm. At this stage, two indicators of the total means of the objective function and total computing time means were used. The result of the calculations indicated the high efficiency of the League Champions algorithm with desirability weight of 0.9516. The following suggestions for future research are:

1. Using discrete algorithms rather than continuous ones
2. Considering more than two machines in problem solving
3. Taking into account buffer objective function as a new objective function
4. Considering uncertainty in the cost of resources assigned to jobs

References

Aldowaisan, T., & Allahverdi, A. (2003). New heuristics for no-wait flowshops to minimize makespan. *Computers & Operations Research*, *30*(8), 1219-1231.

Allahverdi, A., & Aldowaisan, T. (2004). No-wait flowshops with bicriteria of makespan and maximum lateness. *European Journal of Operational Research*, *152*(1), 132-147.

Asadzadeh, L., & Zamanifar, K. (2010). An agent-based parallel approach for the job shop scheduling problem with genetic algorithms. *Mathematical and Computer Modelling*, *52*(11-12), 1957-1965.

Atashpaz-Gargari, E., & Lucas, C. (2007, September). Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In Evolutionary computation, 2007. CEC 2007. IEEE Congress on (pp. 4661-4667). IEEE.

Baker, K. R. (1974). Introduction to sequencing and scheduling, John Wiley and sons Inc. *New York*.

Bai, D., Tang, M., Zhang, Z. H., & Santibanez-Gonzalez, E. D. (2018). Flow shop learning effect scheduling problem with release dates. *Omega*, *78*, 21-38.

Chen, X., Chau, V., Xie, P., Sterna, M., & Błażewicz, J. (2017). Complexity of late work minimization in flow shop systems and a particle swarm optimization algorithm for learning effect. *Computers & Industrial Engineering*, *111*, 176-182.

Chen, J. C., Wu, C. C., Chen, C. W., & Chen, K. H. (2012). Flexible job shop scheduling with parallel machines using Genetic Algorithm and Grouping Genetic Algorithm. *Expert Systems with Applications*, *39*(11), 10016-10021.

Chen, J. S., Pan, J. C. H., & Lin, C. M. (2008). A hybrid genetic algorithm for the re-entrant flow-shop scheduling problem. *Expert systems with applications*, *34*(1), 570-577.

Choi, S. W., & Kim, Y. D. (2008). Minimizing makespan on an m-machine re-entrant flowshop. *Computers & Operations Research*, *35*(5), 1684-1696.

Dempster, M. A. (Ed.). (2012). Deterministic and Stochastic Scheduling: Proceedings of the NATO Advanced Study and Research Institute on Theoretical Approaches to Scheduling Problems held in Durham, England, July 6–17, 1981 (Vol. 84). Springer Science & Business Media.

Engin, O., & Güçlü, A. (2018). A new hybrid ant colony optimization algorithm for solving the no-wait flow shop scheduling problems. *Applied Soft Computing*, *72*, 166-176.

Eren, T., & Güner, E. (2008). A bicriteria flowshop scheduling with a learning effect. *Applied Mathematical Modelling*, *32*(9), 1719-1733.

Gao, F., Liu, M., Wang, J. J., & Lu, Y. Y. (2018). No-wait two-machine permutation flow shop scheduling problem with learning effect, common due date and controllable job processing times. *International Journal of Production Research*, *56*(6), 2361-2369.

Goldbogen, J. A., Friedlaender, A. S., Calambokidis, J., McKenna, M. F., Simon, M., & Nowacek, D. P. (2013). Integrative approaches to the study of baleen whale diving behavior, feeding performance, and foraging ecology. *BioScience*, *63*(2), 90-100.

Hall, N. G., & Sriskandarajah, C. (1996). A survey of machine scheduling problems with blocking and no-wait in process. *Operations research*, *44*(3), 510-525.

He, H. (2016). Minimization of maximum lateness in an m-machine permutation flow shop with a general exponential learning effect. *Computers & Industrial Engineering*, *97*, 73-83.

Kashan, A. H. (2014). League Championship Algorithm (LCA): An algorithm for global optimization inspired by sport championships. *Applied Soft Computing*, *16*, 171-200.

Kashan, A. H., & Karimi, B. (2010, July). A new algorithm for constrained optimization inspired by the sport league championships. In *Evolutionary Computation (CEC), 2010 IEEE Congress on* (pp. 1-8). IEEE.

Laha, D., & Gupta, J. N. (2016). A Hungarian penalty-based construction algorithm to minimize makespan and total flow time in no-wait flow shops. *Computers & Industrial Engineering*, *98*, 373-383.

Li, X., Yang, Z., Ruiz, R., Chen, T., & Sui, S. (2018). An iterated greedy heuristic for no-wait flow shops with sequence dependent setup times, learning and forgetting effects. *Information Sciences*, *453*, 408-425.

Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software*, *95*, 51-67.

Shahvari, O., & Logendran, R. (2018). A comparison of two stage-based hybrid algorithms for a batch scheduling problem in hybrid flow shop with learning effect. *International Journal of Production Economics*, *195*, 227-248.

Shao, W., Pi, D., & Shao, Z. (2017). Optimization of makespan for the distributed no-wait flow shop scheduling problem with iterated greedy algorithms. *Knowledge-Based Systems*, *137*, 163-181.

Tavakkoli-Moghaddam, R., Rahimi-Vahed, A., & Mirzaei, A. H. (2007). A hybrid multi-objective immune algorithm for a flow shop scheduling problem with bi-objectives: weighted mean completion time and weighted mean tardiness. *Information Sciences*, *177*(22), 5072-5090.

Vahedi-Nouri, B., Fattahi, P., & Ramezanian, R. (2013). Minimizing total flow time for the non-permutation flow shop scheduling problem with learning effects and availability constraints. *Journal of Manufacturing Systems*, *32*(1), 167-173.

Ye, H., Li, W., & Miao, E. (2016). An effective heuristic for no-wait flow shop production to minimize makespan. *Journal of Manufacturing Systems*, *40*, 2-7.

Ye, H., Li, W., Abedini, A., & Nault, B. (2017). An effective and efficient heuristic for no-wait flow shop production to minimize total completion time. *Computers & Industrial Engineering*, *108*, 57-69.