Research Article

Parallel machines scheduling with time-dependent deterioration, using meta-heuristic algorithms

Jaber Kalaki Juybari[1] · Somayyeh Kalaki Juybari[2] · Reza Hasanzadeh[1]

Received: 28 June 2020 / Accepted: 3 February 2021 / Published online: 15 February 2021

Abstract
In this paper, we consider the identical parallel machines scheduling problem with exponential time-dependent deterioration. The meaning of time-dependent deterioration is that the processing time of a job is not a constant and depends on the scheduled activities. In other words, when a job is processed later, it needs more processing time compared to the jobs processed earlier. The main purpose is to minimize the makespan. To reach this aim, we developed a mixed integer programming formulation for the problem. We solved problem in small scale using GAMS software, while due to the fact that in larger scales the aforesaid case is a complex and intricate optimized problem which is NP-hard, it is not possible to solve it by standard calculating techniques (in logical calculating times); we applied the meta-heuristic genetic algorithm, simulating annealing and artificial immune system, and their performance has been evaluated. In the end, we showed that solving the problem in small scale, with the meta-heuristic algorithms (GA, SA, and AIS) equals the optimal solution (GAMS), And on a large scale, at a time of approximately equal solution, meta-heuristic algorithm simulating annealing, provides a more optimal solution.

Keywords Scheduling · Parallel machines · Deterioration · Genetic algorithm · Simulating annealing · Artificial immune system

1 Introduction

The basic model of scheduling theory assumes that the processing time of a job is predetermined and constant. However, this assumption may not always be true because of the tools and machinery face deterioration which decreases the machining quality over the time [6]. Parts with turning process or cutting process are another practical example in this field. For example, there are some products that need to be processed by a cutting tool. Because of wear of the cutting tool, the time required for processing a single product increases with respect to the processing time of products already executed [32, 36].

In recent years, the phenomenon known as deterioration has been considered by many researchers, after the pioneering works of Gupta et al. [12] on scheduling models in which they focused on single-machine scheduling with deteriorating jobs and introduced the processing time of a job as the polynomial function of its setup time. Alidaee and Womer [1] have classified scheduling models with deteriorating jobs into three types of linear deterioration, piecewise linear deterioration and nonlinear deterioration. In these types, the actual job processing time directly or indirectly depends on the setup time. In the following, we will review the literature on the related studies.

Chen [5] and Woeginger [37] studied the scheduling problem of deteriorating jobs with the objective function

✉ Jaber Kalaki Juybari, jaber.kalaki@gmail.com; ✉ Reza Hasanzadeh, rhz.1974@gmail.com; Somayyeh Kalaki Juybari, somayyehkalaki@gmail.com | [1]Department of Industrial Engineering, Rouzbahan Institute of Higher Education, Sari, Iran. [2]Department of Food Industry Engineering, Pishva-Varamin Branch, Islamic Azad University, Varamin, Iran.

of the jobs with preemption assuming that the due dates are the same. To solve the problem, they proposed the dynamic programming algorithm with the complexity $O\left(n^2\right)$ and $O\left(n^3\right)$, respectively. In their research, Ji et al. [15] focused on the makespan and the total completion time of deteriorating jobs on machine with the availability constraint while they consider non-resumable case. They showed that both problems are non-deterministic polynomial hard (NP-hard) and present pseudo polynomial time optimal algorithms to solve them. Furthermore, for the makespan problem, present an optimal approximation algorithm and for the total completion time problem, provide a heuristic and evaluate its efficiency by computational experiments. Wang et al. [33] stated that flow shop scheduling with the objective of minimizing makespan in the basic form is NP-hard. They focused on the aforesaid problem in two sections of their study. They first examined it in four different forms and put them in an optimal order. In the second part, a Branch and Bound (B&B) approach was employed to solve the cases with 14 jobs. Moreover, they proposed a heuristic two-stage algorithm which was proved to be effective. Lee and Wu [19] considered the scheduling problem of deteriorating jobs on parallel machines with machines availability constraint. They proposed a new algorithm using the LDR (largest deterioration rate first) rule and the idea of the MULTIFIT heuristic proposed by Coffman et al. [8], for the makespan problem. Mosheiov [23] investigated the makespan in the single-machine cases in which the actual processing time of each job is defined by an exponential function of its position. He also showed that an optimal schedule is a V-shaped schedule in terms of the basic or normal processing times of jobs. In their survey, Lee et al. [20] assumed that the actual processing time of a job in position r is dependent on the processing time of the prior job $r-1$. They demonstrated that longest processing time first rule (LPT) would optimize total completion time problem in a single machine assuming that the deterioration coefficient was between 0 and 1. Ng et al. [24] considered a two-machine flow shop scheduling problem to minimize the total completion time with proportional linear deterioration. They derived it from several dominance properties, some lower bounds, and an initial upper bound, then applied them in an assumed branch-and-bound algorithm to search for the optimal solution. Wang et al. [35] considered the single-machine scheduling problem with learning effect of a time-dependent deterioration. They formulated the model and proposed a mixed integer programming formulation (MIP) for the aforesaid problem and applied the shortest processing time (SPT) rule as a heuristic algorithm for general cases and analyzed its worst-case error bound. Two heuristic algorithms HA1 (is adopted from Kanet [17] idea) and HA2 (Proposed algorithm according to the HA1) which utilize the V-shaped property for the problem of the smallest total completion time were also proposed. Joo and Kim [16] considered the time-dependent deterioration problem and some scheduling problems with rate-modifying activities (i.e. getting recovered to a normal processing time known as RMA). They used a mathematical model to find an optimal solution for minimizing the makespan. Also they considered genetic algorithm with the special character (GA-SC), a genetic algorithm using chromosomes with double string (GA-DS) and a genetic algorithm compounded with dispatching rule (GA-DR). They finally showed that GA-DR provides the highest quality performance, in both effectiveness and efficiency. Torres [31] considered an unrelated parallel machine scheduling problem with time-dependent deterioration given that single-machine problem could be solved in polynomial time. The problem is NP-hard when the scale of the problem grows. They employed a simulated annealing (SA) meta-heuristic algorithm to minimize the total completion time and showed the effectiveness of this algorithm by solving a large number of benchmark instances. Chung and Kim [7] considered a hybrid genetic algorithm (GA) together with a heuristic algorithm for single-machine scheduling problem with time-dependent deterioration and rate-modifying activities. First, they wrote a mixed integer program for optimizing the total completion and applied the problem in scales greater than the hybrid genetic scale with chromosome (GA-SC, GA-DS) and a heuristic solution. They showed that the obtained results are precisely similar.

Shin et al. [28] presented a heuristic and tabu search (TS) algorithm for the problem of single-machine scheduling with sequence-dependent setup time and the release time and the objective function with the aim of minimizing the maximum lateness. Stecco et al. [29] developed a tabu search algorithm for the time-dependent and sequence-dependent single-machine scheduling problem. Bahalke et al. [2] proposed some hybrid meta-heuristic algorithm for single-machine scheduling with sequence dependent on setup time and job deterioration to minimize the makespan. Rabani et al. [26] considered a new hybrid meta-heuristic approach to optimization of parallel machine scheduling problem with human resiliency engineering. They proposed a hybrid meta-heuristic algorithm GA and SA for the problem of non-identical parallel machines and stated that hybrid approach is better than other approaches. Salehi Mir et al. [27] considered scheduling parallel machine problem under general effects of deterioration and learning with past-sequence-dependent setup time with heuristic and meta-heuristic approaches. For solving this problem, they proposed a mixed integer programming model, then used three methodologies including a heuristic algorithm (HA), a genetic algorithm

(GA) and an ant colony optimization (ACO), with a new stochastic result-oriented strategy. They finally showed that the proposed algorithm provides the optimal solution to the small-size problems and the ACO meta-heuristic algorithm statistically better than HA and GA. Cheng et al. [6] studied a class of machines scheduling problems in which the processing time of a job dependent on the setup time. They classified the scheduling problem with different types of time-dependent deterioration and provided a framework for the models of classic problems that had been generalized from the theory of classic scheduling. Wang et al. [32, 36] considered the single-machine scheduling problems with deterioration jobs and group technology assumption. They showed that minimizing the makespan in polynomial time can be solved if the total linear deterioration and group technology are regarded to be simultaneous actions. Wang et al. [34] considered the sing-machine problems of scheduling jobs with start time increasing processing times. The two objectives of the scheduling problems are to minimize the makespan and the total weighted completion time, respectively. Under the assumption of series–parallel graph precedence constraints, they proved that the problems were polynomially solvable.

In the present paper, we have considered the scheduling problem of minimizing the maximum completion time ($C_{\max}$) in parallel and identical machines with a time-dependent deterioration proposed in the studies of Wang et al. [35]. They calculated the actual processing time of the job using the previous scheduled jobs. One of the aims of this study that not observed in previous studies is that of applying scheduling with a time-dependent deterioration Model (in the study of Wang et al. [35]) in the identical parallel machines environment. We developed a mixed integer programming formulation for the problem, then solved it with three meta-heuristic algorithms (GA), (SA) and artificial immune systems (AIS) for the first time and compared the results obtained.

This paper has been organized as follows: in Sect. 2, we wrote a mixed integer programming model. In Sect. 3, as the problem $P_m||C_{\max}$ is strongly NP-hard for higher numbers of m [11], we solved the problem in a larger scale using the meta-heuristic algorithms (genetic, simulating annealing, artificial immune system) and evaluated their performance. The final section dealt with concluding remarks.

2 Model formulation

Consider the identical parallel machines problem. There is a set of n jobs $i = \{i_1, i_2, \dots, i_n\}$ where all jobs are available for processing at time 0 and setup time = 0. The machine can handle one job at a time and preemption is not allowed. Each job is associated with a normal processing time p_i and the normal processing time of a job if scheduled at the rth position in a sequence $p_{[r]}$. Suppose that the p_{ir} is the processing time of job i when it is scheduled at the position r in a sequence. Like the study of Wang et al. [35], we consider a time-dependent deterioration model using Eq. (1):

$$p_{ir} = p_i\left(1 + p_{[1]} + p_{[2]} + \cdots + p_{[r-1]}\right)^a \tag{1}$$

In this relation, the actual processing time in position r depends on the basic processing time of the previous scheduled activity $r - 1$. Where $a \geq 0$ is the deterioration index.

According to the three-field notation scheme $\alpha|\beta|\gamma$ introduced by Graham et al. [13], we show the problem considered in this research as $P_m|p_{ir} = p_i\left(1 + p_{[1]} + p_{[2]} + \ldots + p_{[r-1]}\right)^a,\ 0 < a < 1|C_{\max}$.

Notations

m: The number of available machines.

n: The number of jobs which need to be scheduled.

j: The index of machines, $j = 1, \dots, m$.

i: The index of jobs, $i = 1, \dots, n$.

r: The index of the position of each job, $r = 1, \dots, n$.

Parameters

P_i: The normal processing time of job i.

t_j: The setup time of the first scheduled job in machine j.

α: The deterioration index.

Decision variables

X_{irj}: Equals 1 when job i is done in position r on machine j. Otherwise, it is 0.

$P_{[r]j}$: The actual processing time of the job scheduled in position r on machine j.

$C_{[r]j}$: The actual completion time the job scheduled in position r on machine j.

C_{max}: The total completion time (completion time of the machine with the maximum load,$C_{max} = \max\{C|j = 1, 2, \dots, n\}$).

Based on the parameters and decision variables, a mixed integer programming process can be formulated.

Constraints

Constraint (2) stipulates that each job must be processed.

$$\sum_{j=1}^{m}\sum_{r=1}^{n} X_{irj} = 1 \quad \forall i = 1, \dots, n. \tag{2}$$

Constraint (3) guarantees that just one job be processed in any position on any machine.

$$\sum_{i=1}^{n} X_{irj} \leq 1 \quad \forall j = 1, \dots, m; r = 1, \dots, n. \tag{3}$$

Constraint (4) guarantees that immediately after each job, just one of the remaining jobs can be done.

$$X_{irj} + \sum_{l=1, l\neq i}^{n} X_{lr+1j} \leq 2 \quad \forall i = 1,\dots,n; j = 1,\dots,m; r = 1,\dots,n-1. \tag{4}$$

Constraint (5) shows that when job *i* is not in its initial position, another job must be done immediately before job *i*.

$$X_{i,r+1,j} \leq \sum_{l=1, l\neq i}^{n} X_{lrj} \quad \forall i = 1,\dots,n; j = 1,\dots,m; r = 1,\dots,n-1. \tag{5}$$

Constraint (6) shows the processing time of the job scheduled in the first position on machine *j*.

$$P_{[1]j} = \sum_{i=1}^{n} (P_i) X_{i,1,j} \quad \forall j = 1,\dots,m. \tag{6}$$

Constraint (7) shows the processing time of the job scheduled in position *r* on machine *j*.

$$P_{[r]j} = \sum_{i=1}^{n} (P_i) X_{i,r,j} \quad \forall j = 1,\dots,m; r = 2,\dots,n. \tag{7}$$

Constraints (8) and (9) show the actual completion time of the job scheduled in the first and *r*th position on machine *j*.

$$C_{[1]j} = t_j + P_{[1]j} \quad \forall j = 1,\dots,m, \tag{8}$$

$$C_{[r]j} = C_{[r-1]j} + P_{[r]j}\left(1 + \sum_{i=1}^{r-1} P_{[i]j}\right)^{\alpha} \quad \forall j = 1,\dots,m; r = 2,\dots,n. \tag{9}$$

Constraint (10) shows the upper limit $C_{\max}$.

$$C_{[r]j} \leq C_{\max} \quad \forall j = 1,\dots,m; r = 1,\dots,n. \tag{10}$$

Constraint (11) shows the range decision variables.

$$X_{irj} = 0, 1, P_{[r]j}, C_{[r]j}, C_{\max} \geq 0 \quad \forall i = 1,\dots,n; j = 1,\dots,m; r = 1,\dots,n. \tag{11}$$

Objective function

The objective of the scheduling problem in parallel machines is to minimize the completion time of the machine with the maximum load which is shown in Eq. (12).

$$Min C_{\max} \tag{12}$$

3 Meta-heuristic algorithms

We propose three effective and efficient meta-heuristic algorithms, because the mixed integer programming model is not suitable for the problems over 10 jobs because of the long computation time [16]. The parallel machines scheduling with exponential time-dependent deterioration considered in this paper is typical combinatorial optimization problem, and GA, SA and AIS are known as effective and efficient algorithms for combinatorial optimization problems [3, 16, 31].

3.1 Genetic algorithm

Genetic algorithm first introduced by Holland [14] is one of the evolutionary optimization methods that inspired by the laws of evolutionary biology such as inheritance, biology mutation and Darwin's principles of choice. Natural selection is the process through which organisms that are more adapted to their environment are more likely to survive and reproduction and can pass on their genes to the next generation.This algorithm starts with an initial set of solutions (initial population). Each member of the population is called a chromosome that defines a solution to the problem. Chromosomes are evaluated based on the performance criterion and are evolved through continuous repetition. To produce each generation, the new chromosomes are produced by combination (the crossover operator) or alteration (the mutation operator) of the previous generation chromosomes. This process continues until the stopping criterion is met. In the following paragraph, we have briefly explained the steps of the proposed algorithm.

1. Arranging the genetic algorithm parameters.
2. Producing and evaluating the randomly selected initial population.
3. Selecting the individuals and the parents from the population to implement the mutation and crossover operators.
4. Combining the actual population, the offspring population and the population of mutated ones to choose the actual population of the new generation.
5. Stopping at values of the predetermined repetitions.

In the present study, parents have been selected based on the Roulette wheel selection (fitness proportionate selection). It is one of the operators used in GA. In fitness proportionate selection, as in all selection methods, the fitness function assigns a fitness level to possible solutions or chromosomes. This fitness level is used to assign a probability of selection to each individual chromosome. If f_i is the fitness of individual *i* in the population, its probability of being selected is equal to relation (13).

$$P_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \tag{13}$$

where N is the number of individuals in the population [4]. Moreover, we applied elitism method in which the best chromosomes are moved from the previous stages to the present stage.

3.2 Simulated annealing algorithm

Simulated annealing algorithm (SA) is a kind of neighborhood search-based meta-heuristic algorithm which uses probabilistic and mathematical to find the global optimum. In other words, this algorithm attempts to approximate global optimization in a large space. The name of the aforesaid algorithm comes from annealing process in metallurgy which involves heating and then controlled cooling of a material to increase the size of its crystals. For the first time, this algorithm was introduced by Kirkpatrick et al. [18]. The main advantage of simulated annealing is its ability to avoid getting stuck at the local minimum. This approach not only accepts changes that improve the objective function, but also accepts some changes, with a certain probability, that do not improve the objective function solution in order not to get caught up in a local solution.

As a whole, the following phases were used in the algorithm SA:

1. Define the initial configuration set S.
2. Set the initial temperature $T > 0$.
3. If stop criterion invalid,
    - 3.1 Execute the following loop
    - 3.2 Select a random neighborhood from S and call it S'
    - 3.3 Let $\Delta = \text{cost}\, S' - \text{cost}\, S$
    - 3.4 If $\Delta \leq 0$ then $S' = S$.
    - 3.5 If $\Delta > 0$ then $S' = S$ with a probability $e^{-\left(\frac{\Delta}{T}\right)}$.
    - 3.6 T as a controlling parameter. It is small, however, close to 1.
4. Return to S.

In the simulated annealing algorithm, in addition to the main loop of the algorithm, we defined an internal loop for the algorithm. It is clear that this will only reduce the temperature of the system after the inner ring is completed. Therefore, the search algorithm increases at a constant temperature and the probability of discovering better points increases.

3.3 Artificial immune system algorithm

Like the genetic algorithm, artificial immune system algorithm is inspired by natural behavior in the real world. It was first introduced by De Castro et al. [9]. This algorithm is inspired by the biological immune system in dealing with pathogens. It is similar to the genetic algorithm in many aspects; however, there are differences in the process of the algorithm that enable this algorithm to be faster to converge and to need lower number of calculations. The phases of artificial immune system algorithm are as follows:

1. Setting a randomly selected initial population of antibodies.
2. Evaluating each antibody by the objective function.
3. Selecting the best antibodies to perform reproduction.
4. Performing the reproduction for each selected antibody.
5. Implementing the mutation operator on the reproduced antibodies based on the level of overlap between them (the objective function).
6. Replacing the worst antibodies in the reference population with the best ones in the mutated population.
7. Return to 2, until the stop condition is reached.

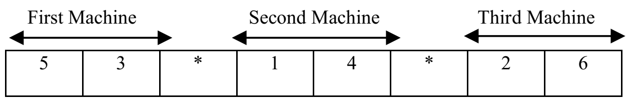


Fig. 1 Chromosome representation with the special character

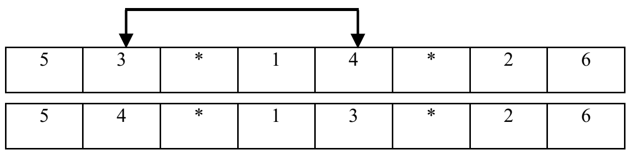


Fig. 2 Swap mutation

3.4 Chromosome representation

According to Fig. 1, representation of chromosomes in three algorithms is in form of a special character.

We have used swap and reversion mutations to implement the mutation operator in genetic algorithm, generate neighborhood solution in the simulated annealing algorithm and mutation on antibodies in artificial immune system algorithm. According to Fig. 2, in the swap mutations, first a chromosome is randomly selected. Then two genes of the chromosome are selected randomly and their places are exchanged to implement the mutation operator. Moreover, according to Fig. 3, in the reversion mutation, first two points of a chromosome are randomly selected and the genes between these two points are reversed.

According to Fig. 4, in order to implement the crossover operator in GA of generation A, two chromosomes were

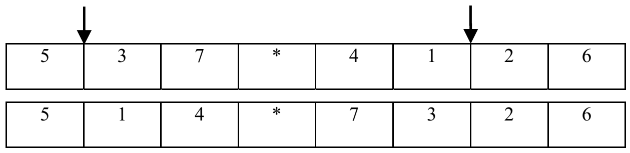


Fig. 3 Reverse mutation

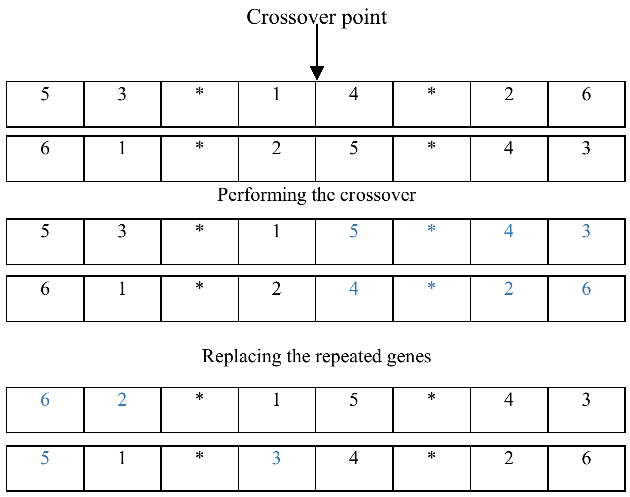


Fig. 4 One-point crossover

randomly selected and the single-point crossover was implemented on them to generate the offspring population. To do this, we first perform the crossover operation and then by keeping the second part of the child's chromosomes constant, delete the repeated genes in the first part and put the values never used in the chromosome in the blank spots.

4 Computational results

Based on past and extensive experiences in this field, in this study we adjusted the parameter levels so that the algorithms are optimized at a relatively equal time. Accordingly, we investigated the effective parameters of GA, SA and AIS algorithms using Taguchi method in Minitab software.

The method of designing Taguchi experiments was introduced in 1960 by Professor Taguchi [22]. Taguchi developed a new experimental design method to increase the efficiency of implementation and evaluation of experiments. Its structure is more suitable for evaluating production processes because the required number of experiments is reduced significantly. The design of experiment using the Taguchi method provides a simple, efficient and systematic approach to determine optimum conditions [10]. Taguchi method offers orthogonal arrays as a mathematical tool that analyses the smallest number of experiments that have a large number of parameters. Thus, it would reduce time and effort. In this method, experimental results are converted to a signal/noise (S/N) ratio which means a ratio of an average standard deviation [30]. A larger S/N ratio indicates a better test result. So in experiments, a level of the factor which has the highest ratio represents a better performance. The ratio allows controlling mean and variance while at the same time an analysis of variance (ANOVA) is performed. In this way, the effects of factors can be revealed statistically. S/N ratio can be calculated with Eq. (14):

$$S/N\,\text{Ratio} = -10\,\text{Log}_{10}\left[\frac{1}{n} \times \sum_{i=1}^{n} y^2\right] \tag{14}$$

n, the number of observations in the experiment and y_i makespan values ($i = 1$ ton).

The parameter levels are presented in Table 1 for all three algorithms. For the experiments, we considered 100 job and 4 machines with a degradation rate of 0.45. The processing time of the jobs were randomly selected from the intervals [1, 50]. And an average of 30 times run were used for each experiment. To get the exact average of each. According to the number of parameters and levels, for each of the GA, SA and AIS algorithms, 9 different experiments are performed according to the table

Table 1 Factors and test levels GA, SA and AIS

Algorithm	GA		SA		AIS	
Parameter	A: [P_c, P_m]	B: [N_{pop}, Itr]	A: [Alfa, T_0]	B: [Itr, Itr_1]	A: N_a	B: [Itr, N_c]
Level 1	0.8, 0.2	60, 200	0.90, 30	200, 50	15	200, 6
Level 2	0.75, 0.25	80, 150	0.98, 50	150, 70	20	100, 8
Level 3	0.85, 0.15	100, 100	0.95, 70	100, 100	30	50, 10

P_c: Crossover operator, P_m: Mutation operator, N_{pop}: Initial population of chromosomes, *Itr*: The number of iterations of the algorithm, *Alfa*: Temperature reduction coefficient,T_0: Initial temperature, *Itr*1: The number of repetitions of the inner loop, N_a: The number of antibodies population, N_c: The number of antibodies that have most compatibility with the antigen.

(orthogonal), i.e. in total, for each of the algorithms, $9 \times 30 = 270$ separate experiments were performed.

According to Fig. 5, we considered for GA the initial population of 60, the number of iterations 200, the crossover and mutation rates of 0.75 and 0.25.

According to Fig. 6, we considered for SA the iterations of the main loop 200, the inner loop iterations 50, the initial temperature 30 and the temperature reduction coefficient 0.90.

According to Fig. 7, we considered for AIS the number of iterations is 200, the number of antibodies 20, and the number of antibodies that have the highest compatibility with the antigen 6. We performed the numerical experiments on a computer with Intel core i3, 1.8 GHz and RAM 4 GB. The processing time of the jobs were randomly selected from the intervals [1, 50].

In this paper, the quality of the answers obtained from the meta-heuristic algorithms has been evaluated by the relative percent deviation (RPD) from the best-known solutions. RPD is a criterion for unsealing data. The reason for using RPD is to normalize the outputs for comparison. A lower RPD RPD value indicates a more optimal answer which is calculated by expression (15).

$$RPD = \frac{\text{method}_{sol} - \text{Best}}{\text{Best}} \times 100 \tag{15}$$

where method_{sol} equals the value of the objective function obtained from the meta-heuristic (GA, SA and AIS)

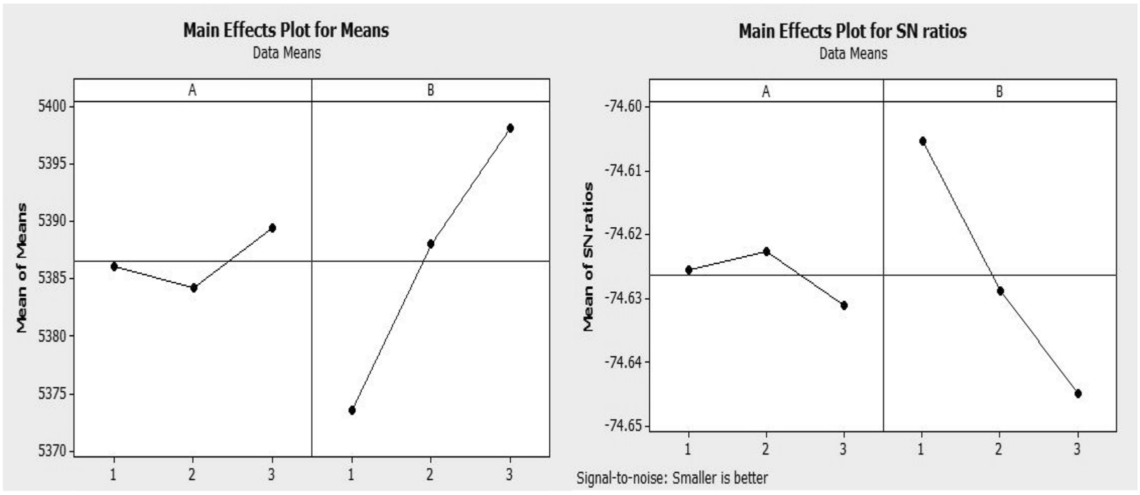


Fig. 5 Mean and S/N ratio diagram for GA

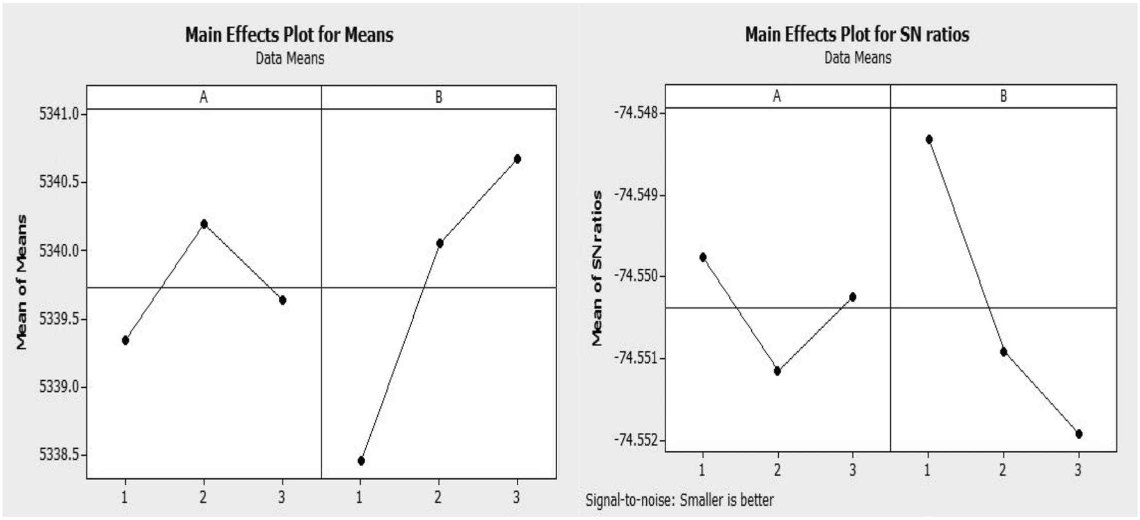


Fig. 6 Mean and S/N ratio diagram for SA

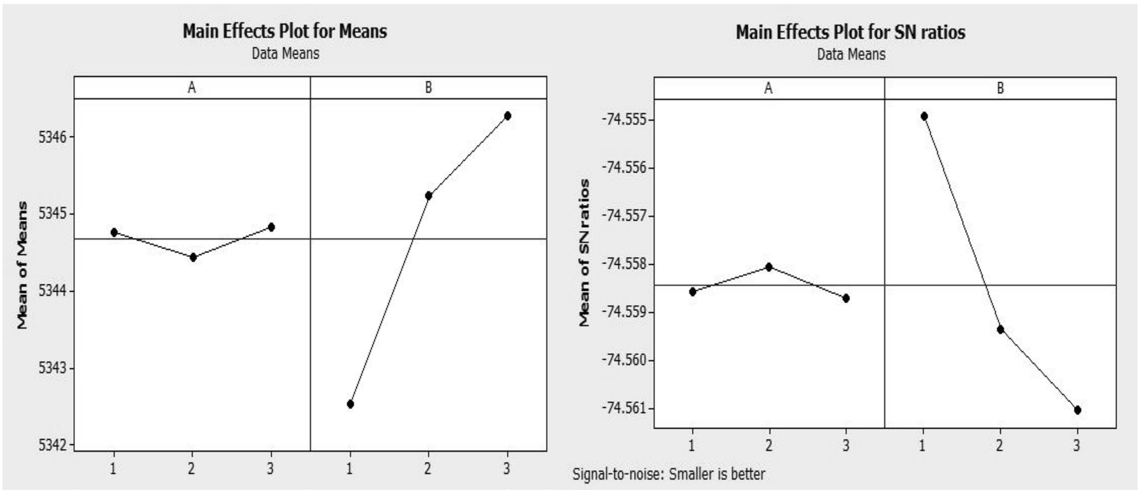


Fig. 7 Mean and S/N ratio diagram for AIS

solution and (Best) equals the optimized answer obtained from the model solution by general algebraic modeling system (GAMS) software in the certain form. (GAMS is a high-level modeling system for mathematical optimization and designed for modeling and solving linear, nonlinear, and mixed integer optimization problems. The system is tailored for complex, large-scale modeling applications.) Moreover, in the following tables, m is the number of the machines, n is the number of jobs, and α is the deterioration rate. GA, SA and AIS in Table 3 are twice as large as the best obtained value and mean equals their means in 30 times of RUN in three algorithms. MAD stands for the mean absolute deviation.

In Table 2, we evaluated the performance of the algorithms (GA, SA and AIS) in small scales ($m = 3; n = 7, 8, 9$ and $m = 4; n = 7, 8$ and according to the study Wang et al. [35] $\alpha = 0.05, 0.45, 0.85$) in comparison with the optimized answer obtained from the optimized solution (GAMS) of the total number of 15 problems for each methods.

The zero RPD index indicates that all three considered meta-heuristic algorithms are efficient for this problem. In Table 3, we evaluated three algorithms of GA, SA and AIS in average scale ($m = 3; n = 30, 40; \alpha = 0.05, 0.45$) and in large scale ($m = 4, 5, 6; n = 100, 150, 200; \alpha = 0.05, 0.45$) performing the algorithm 30 times for each problem and 420 problems for each of the algorithms (GA, SA and AIS).

Table 2 Test results of small-sized problems

m	n	α	Opt	Time (s)	GA		SA		AIS	
					RPD	Time (s)	RPD	Time (s)	RPD	Time (s)
3	7	0.05	35.005	523	0.00	5.21	0.00	5.08	0.00	5.11
		0.45	81.781	643	0.00	5.22	0.00	5.01	0.00	5.11
		0.85	232.037	870	0.00	5.21	0.00	5.11	0.00	5.15
	8	0.05	39.674	621	0.00	5.55	0.00	5.18	0.00	5.19
		0.45	95.674	902	0.00	5.55	0.00	5.23	0.00	5.21
		0.85	285.568	1022	0.00	5.58	0.00	5.20	0.00	5.23
	9	0.05	45.191	1210	0.00	5.77	0.00	5.22	0.00	5.19
		0.45	114.877	1335	0.00	5.72	0.00	5.24	0.00	5.25
		0.85	368.529	1470	0.00	5.69	0.00	5.22	0.00	5.21
4	7	0.05	27.522	872	0.00	5.30	0.00	5.19	0.00	5.20
		0.45	51.785	896	0.00	5.32	0.00	5.16	0.00	5.17
		0.85	127.143	902	0.00	5.31	0.00	5.25	0.00	5.15
	8	0.05	32.671	1008	0.00	5.51	0.00	5.23	0.00	5.33
		0.45	61.187	1023	0.00	5.56	0.00	5.54	0.00	5.39
		0.85	159.375	1097	0.00	5.59	0.00	5.29	0.00	5.29
Average					0.00		0.00		0.00	

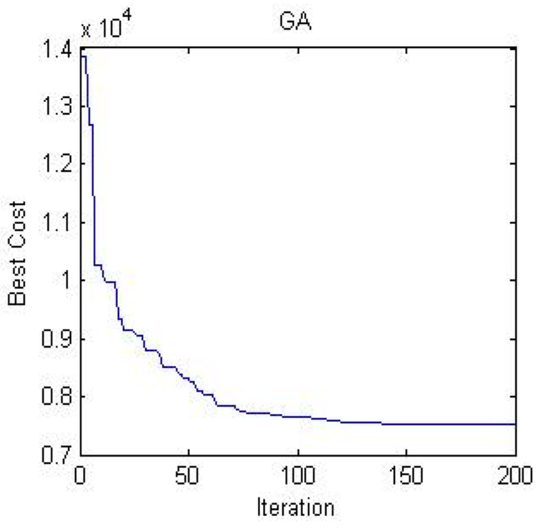
Table 3 Test results of medium and large-sized problems

m	n	α	GA	Time (s)	Mean	MAD	SA	Time (s)	Mean	MAD	AIS	Time (s)	Mean	MAD
3	30	0.05	176.936	13.43	177.565	0.595	176.043	10.68	176.215	0.168	176.063	12.61	177.016	0.280
		0.45	858.958	13.24	863.387	1.782	857.968	10.97	861.316	1.699	858.782	12.54	862.906	1.450
3	40	0.05	299.025	16.42	300.1219	0.601	298.803	13.39	299.084	0.169	298.780	15.37	299.885	0.706
		0.45	1805.363	16.36	1815.367	6.291	1799.957	13.19	1806.372	2.654	1804.255	15.59	1812.792	6.149
4	100	0.05	637.979	33.82	639.246	0.514	636.361	25.51	637.858	0.236	637.427	31.43	638.635	0.612
		0.45	5341.278	34.32	5365.571	7.421	5315.243	25.46	5322.625	3.624	5338.459	31.56	5348.935	6.915
5	100	0.05	504.075	34.41	505.21	0.401	501.873	26.84	502.539	0.275	503.463	31.54	504.231	0.534
		0.45	3823.52	34.25	3840.796	7.956	3798.292	26.9	3803.117	2.739	3807.588	32.18	3823.823	6.897
5	150	0.05	746.734	48.11	747.729	0.446	744.468	34.11	745.169	0.447	746.219	45.14	747.498	0.722
		0.45	6694.974	48.18	6708.089	7.666	6642.251	34.24	6653.157	4.758	6679.131	45.23	6691.918	7.066
6	150	0.05	615.708	49.57	617.185	0.865	613.839	34.12	614.451	0.358	614.999	45.53	616.561	0.819
		0.45	5121.318	48.32	5137.351	7.061	5060.628	35.53	5068.282	5.683	5092.047	45.23	5111.098	7.185
6	200	0.05	809.827	62.12	811.912	0.613	807.498	46.55	808.351	0.502	809.670	58.26	810.764	0.504
		0.45	7516.581	62.65	7528.973	7.939	7459.001	46.58	7466.895	5.101	7505.705	58.44	7517.854	7.339

Solving the problem in medium and large scales shows that in 30 times running the algorithms according to the best solution obtained, the mean of the answers and the values of MAD, SA are more optimum than GA and AIS. Moreover, as we employed the exponential function of deterioration, we witness some differences in the final answers given the deterioration rate and the size of the problem that MAD values indicate this.

According to Figs. 8, 9 and 10, we see that the convergence speed of the SA is higher. In the following, we interpret the results of three algorithms using Wilcoxon rank-sum test in Minitab software. Mann–Whitney U test or Wilcoxon rank-sum test is a nonparametric test that examines the difference between two independent groups regarding a variable with rank or sequential data [21]. In fact, this test is the nonparametric equivalent of the independent t test, but with the difference that the t test is parametric and its data is continuous, while the Mann -Whitney U test is nonparametric and is performed with rank data. The null and two-sided research hypotheses for the nonparametric test are stated as follows:

1. All the observations from both groups are independent of each other,
2. The responses are ordinal (i.e., one can at least say, of any two observations, which is the greater),
3. Under the null hypothesis H_0, the distributions of both populations are equal [25].



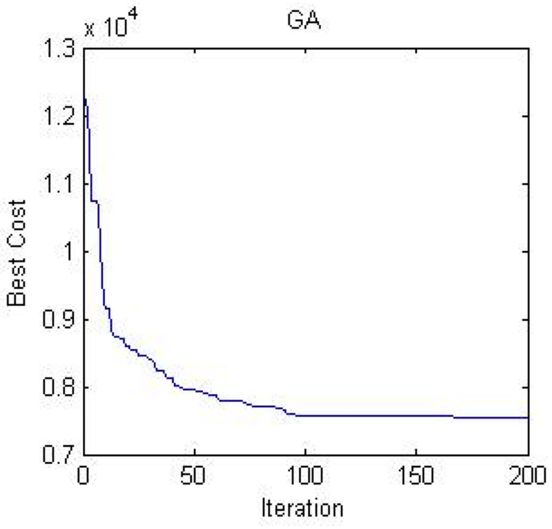


Fig. 8 Convergence diagram of GA with $\alpha = 0.45$, $n = 200$ and $m = 6$

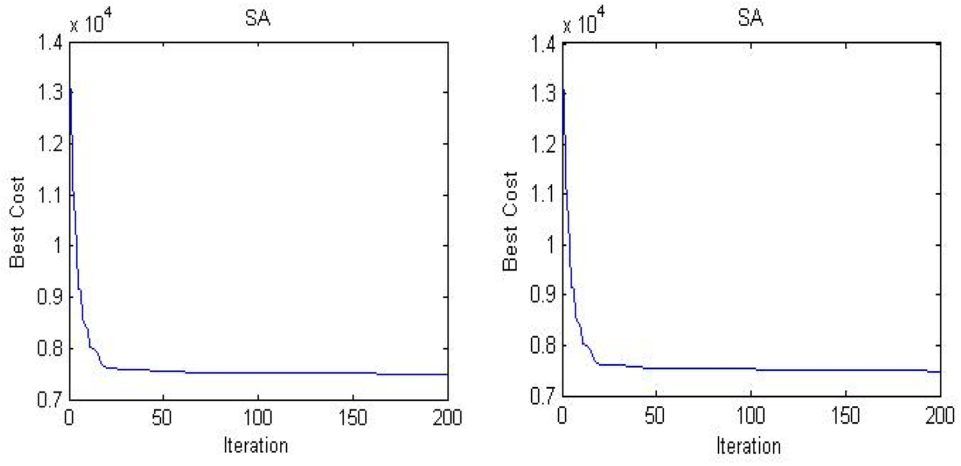


Fig. 9 Convergence diagram of SA with $\alpha = 0.45$, $n = 200$ and $m = 6$

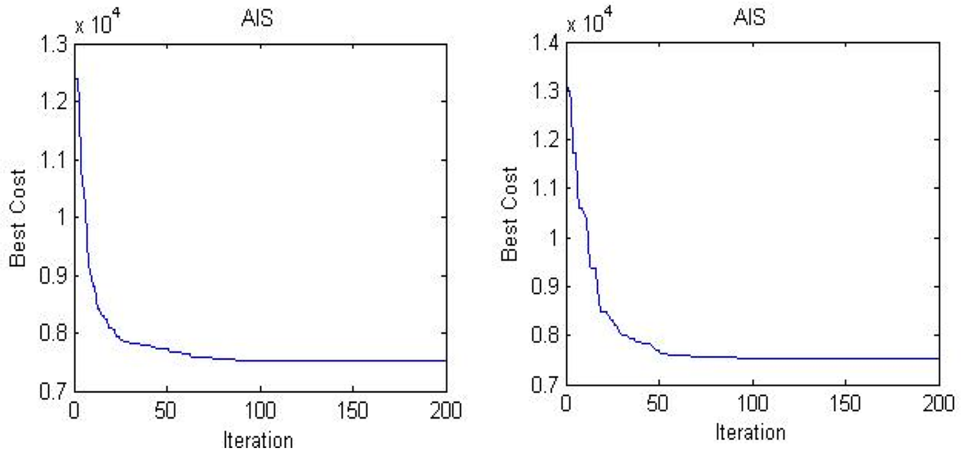


Fig. 10 Convergence diagram of AIS with $\alpha = 0.45$, $n = 200$ and $m = 6$

4. The alternative hypothesis H_1 is that the distributions are not equal.

The Mann–Whitney test statistic is defined as Eqs. (16) and (17), [38]:

$$U_1 = R_1 - \frac{n_1(n_1 + 1)}{2} \tag{16}$$

$$U_2 = R_2 - \frac{n_2(n_2 + 1)}{2} \tag{17}$$

where n_1 and n_2 are the sum of groups 1 and 2 and R_1 and R_2 are the sum of the ranks of groups 1 and 2, respectively. A smaller value between U_1 and U_2 is used for comparison in the test phase. Therefore according to Eq. (18):

$$U = \min\{U_1, U_2\} \tag{18}$$

If the U statistic at the $1 - \alpha$ confidence level is greater than the value obtained from the table, the null hypothesis is not accepted. In the statistical significance test of U, if the smaller group size is 20 items or less and the larger sample size is 40 items or less, the U Mann–Whitney critical value table is used. However, if the volume of both groups is greater than 20 or the volume of one of them is greater than 40, then the distribution of the U statistic tends to the normal distribution. In this case, by calculating the mean and standard deviation, U is calculated and the Z statistic is calculated using Eq. (19).

$$z = \frac{U - m_u}{\sigma_u} \tag{19}$$

where in $m_u = \frac{n_1 n_2}{2}$ Average u $\sigma_u = \sqrt{\frac{n_1 n_2 (n_1 + n_2 + 1)}{12}}$ the standard deviation is U. If the value of Z statistic is greater than the value obtained from the standard normal distribution table for the $1 - \alpha$ confidence level, the null hypothesis is not confirmed. Therefore, in this study, we examined the results of SA with GA and AIS in the confidence interval of 0.95.

Table 4 Wilcoxon rank-sum test results

Sample		N	Descriptive statistics		Estimation for difference		Adjusted for ties	
			Median	Difference	CI for difference	Achieved confidence	W-value	*P* value
SA, GA	SA	14	834.8	− 3.1	(− 1334.2,1287.6)	95.4	196.0	0.3826
	GA	14	837.6					
SA, AIS	SA	14	834.8	− 2.4	(− 1310.4,1301.9)	95.4	196.0	0.3826
	AIS	14	836.8					

N: Sample size, CI: Confidence intervals, W: Wilcoxon W test statistics

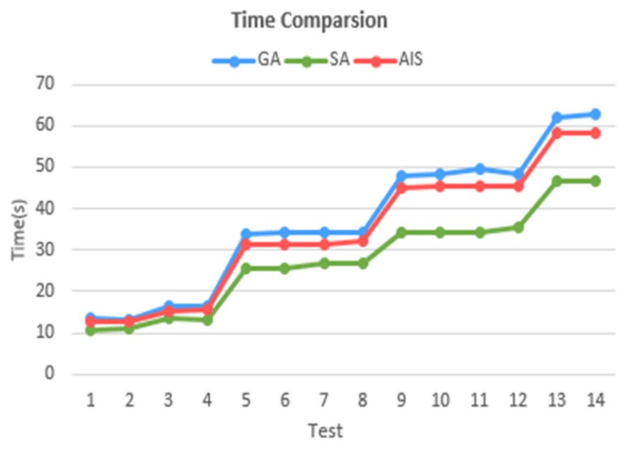


Fig. 11 Comparison of the solution time of GA-SA-AIS method

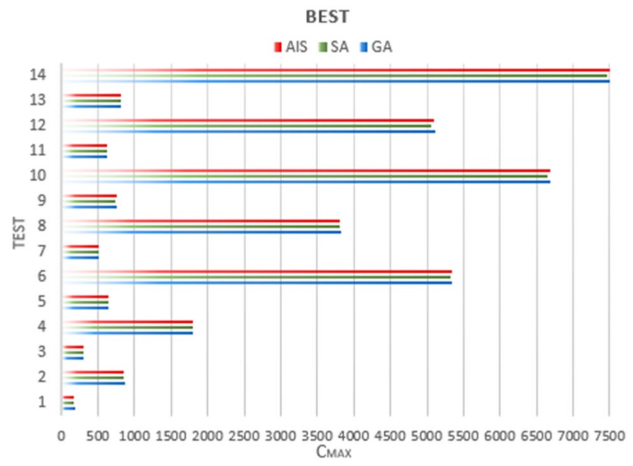


Fig. 12 Comparison of the best answer with 30 times of repetition

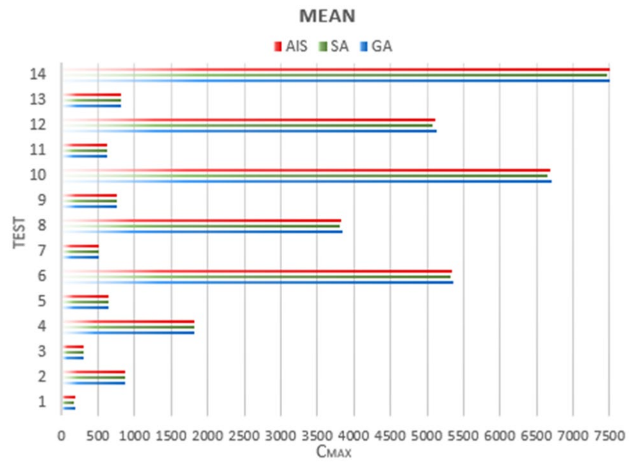


Fig. 13 Comparison of mean values with 30 times of repetition

In the SA, GA test, H_0: The average SA population is lower than GA, and In the SA, AIS test, H_0: The average SA community is less than AIS.

In Table 4, given a significance probability of 0.3826, we cannot reject the null hypothesis. Therefore, SA is less than GA and AIS.

In Fig. 11, we see the problem solving time diagram with the three algorithms GA, SA and AIS, which show that in larger sizes the time to solve SA is less than GA and AIS.

In Fig. 12, we see the most optimal solving values in 30 times the iteration of the solution with the three meta-heuristic algorithms GA, SA and AIS, which shows that the SA method provides more optimal values.

In Fig. 13, we see the mean values of the solutions obtained 30 times in the solution of the three meta-heuristic algorithms GA, SA and AIS. Which shows that the SA method provides more optimal values.

5 Conclusions

In this paper, we considered scheduling problem with exponential time-dependent deterioration in the study of Wang et al. [35] in which they assessed the actual processing time of jobs according to the previous scheduled jobs. They formulated the model and proposed a mixed integer programming formulation for this problem and used two heuristic algorithms which utilize the V-shaped property for the problem of the smallest total completion time and showed that the proposed algorithms provide the optimal solution. Due to the importance of the problem and its implications in different settings, we reached a more effective approach by implementing the aforesaid relation in the setting of identical parallel machines. (Theoretically, the problem of parallel machines can be regarded as the generalization of single-machine problems and a special form of existing scheduling problems in flexible production systems. When it comes to practice, there are also some similar working stations in many production settings that have a lot of similar equipment with similar or different practical characteristics). To solve the problem in small scale, we developed a mixed integer programming model with the objective function C_{max} that improves the optimized use of the machine and the level of efficiency. In order to solve the problem in larger scales, we applied three meta-heuristic algorithms GA, SA and AIS. Finally, after tuning of parameters GA, SA and AIS using Taguchi method and analyzing the results with Wilcoxon rank-sum test, we showed that by expanding the problem scale and the deterioration rate, SA has a higher efficiency than GA and AIS. Future studies can evaluate the effectiveness of other meta-heuristic algorithms related to this problem. Moreover, they can analyze scheduling problem with exponential deterioration in other scheduling environment that embraces a vast area in scheduling problems.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Alidaee B, Womer NK (1999) Scheduling with time dependent processing times. Review and extensions. J Oper Res Soc 50:711–721
2. Bahalke U, Yolmeh AM, Shahanaghi K (2010) Meta-heuristics to solve single machine scheduling problem with sequence-dependent setup-time and deteriorating jobs. Int J Adv Manuf Technol 50:749–759
3. Balaji AN, Porselvi S (2014) Artificial immune system algorithm and simulated annealing algorithm for scheduling batches of parts based on job availability model in a multi-cell flexible manufacturing system. Procedia Engineering 97:1524–1533
4. Blickle T, Thiele L (1996) A comparison of selection schemes used in evolutionary algorithms. Evolut Comput 4:361–394
5. Chen L (1995) A note on single-processor scheduling with time-dependent execution times. Oper Res Lett 17:127–129
6. Cheng TCE, Ding Q, Lin BMT (2004) A concise survey of scheduling with time-dependent processing times. Eur J Oper Res 152:1–13
7. Chung B, Kim B (2016) A hybrid genetic algorithm with two-stage dispatching heuristic for a machine scheduling problem with step-deteriorating jobs and rate modifying activities. Comput Ind Eng 98:113–124
8. Coffman EG, Garey MR, Johnson DS (1978) An application of bin-packing to multi-processor scheduling, SIAM. J Comput 7:1–17
9. De Castro LN, Von Z, Jose F (2001) aiNet: an artificial immune network for data analysis. Idea Group Publishing, Philadelphia
10. Du Plessis BJ, De Villiers GH (2007) The application of the Taguchi method in the evaluation of mechanical flotation in waste activated sludge thickening. Resour Conserv Recycl 50:202–210
11. Garey M, Johanson J (1979) A guide to the theory of NP-complete. Computers and intractability. W.H. Freeman, Sanfrancisco
12. Gupta JND, Gupta SK (1988) Single facility scheduling with non-linear processing times. Comput Ind Eng 14:387–393
13. Graham RL, Lawler EL, Lenstra JK, Rinnooy Kan AHG (1979) Optimization and approximation in deterministic sequencing and scheduling: a survey. Ann Discrete Math 5:287–326
14. Holland JH (1975) Adaptation in natural and artificial systems. An introductory analysis with applications to biology, control, and artificial intelligence. University of Michigan Press, Ann Arbor, MI
15. Ji M, He Y, Cheng TCE (2006) Scheduling linear deteriorating jobs with an availability constraint on a single machine. Theoret Comput Sci 362:115–126
16. Joo C, Kim B (2013) Genetic algorithms for single machine scheduling with time-dependent deterioration and rate-modifying activities. Expert Syst Appl 39:3036–3043
17. Kanet JJ (1981) Minimizing variation of flow time in single machine systems. Manag Sci 27:1453–1459
18. Kirkpatrick S, Gellat CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680
19. Lee WW, Wu CC (2008) Multi-machine scheduling with deteriorating jobs and scheduled maintenance. Appl Math Model 32:362–373
20. Lee WC, Wu CC, Liu HC (2009) A note on single-machine makespan problem with general deteriorating function. Int J Adv Manuf Technol 40:1053–1056
21. Mann HB, Whitney DR (1947) On a test of whether one of two random variables is stochastically larger than the other. Ann Math Stat 18:50–60
22. Mori T (1990) The new experimental design, Taguchi's approach to quality engineering, 1st edn. ASI Press, Dearborn
23. Mosheiov G (2005) A note on scheduling deteriorating jobs. Math Comput Model 41:883–886
24. Ng CT, Wang J-B, Cheng TCE, Liu LL (2010) A branch-and-bound algorithm for solving a two-machine flow shop problem with deteriorating jobs. Comput Oper Res 37:83–90
25. Pratt J (1964) Robustness of some procedures for the two-sample location problem. J Am Stat Assoc 59:655–680
26. Rabani M, Aghamohamadi S, Yazdanparast R (2019) Optimization of parallel machine scheduling problem with human resiliency engineering: a new hybrid meta-heuristics approach. J Ind Syst Eng 12:31–45

27. Salehi Mir MS, Rezaeian J, Mohamadian H (2020) Scheduling parallel machine problem under general effects of deterioration and learning with past-sequence-dependent setup time: heuristic and meta-heuristic approaches. Soft Comput 24:1335–1355
28. Shin HJ, Kim C-O, Kim SS (2002) A tabu search algorithm for single machine scheduling with release times, due dates, and sequence-dependent setup times. Int J Adv Manuf Technol 19:859–866
29. Stecco G, Cordeau J-F, Moretti E (2009) A tabu search heuristic for a sequence dependent and time-dependent scheduling problem on a single machine. J Sched 12:3–16
30. Taguchi G, Chowdhury S, Wu Y (2005) Taguchi's quality engineering handbook. Wiley, Hoboken, NJ
31. Torres A (2013) Parallel machine scheduling to minimize the makespan with sequence dependent deteriorating effects. Comput Oper Res 40:2051–2061
32. Wang JB, Gao W-J, Wang LY, Wang D (2009) Single machine group scheduling with general linear deterioration to minimize the makespan. Int J Adv Manuf Technol 43:146–150
33. Wang JB, Ng CTD, Chen TC (2006) Minimizing total completion time in a two-machine Flow shop with deteriorating jobs. Appl Math Comput 180:185–193
34. Wang JB, Ng CT, Cheng TCE (2008) Single-machine scheduling with deteriorating jobs under a series-parallel graph constraint. Comput Oper Res 35:2684–3269
35. Wang JL, Sun L, Sun L (2011) Single-machine total completion time scheduling with a time-dependent deterioration. Appl Math Model 35:1506–1511
36. Wang JB, Wang IY, Wang D, Wang XY (2009) Single-machine scheduling with a time-dependent deterioration. Int J Adv Manuf Technol 43:805–809
37. Woeginger GJ (2005) Scheduling with time-dependent execution times. Inf Process Lett 48:155–156
38. Zar JH (1998) Biostatistical analysis. Prentice Hall International Inc, New Jersey, p 147