Available online at www.sciencedirect.com
ScienceDirect
Mathematics and Computers in Simulation 180 (2021) 152–177
MATHEMATICS AND COMPUTERS IN SIMULATION
www.elsevier.com/locate/matcom

Original Articles

A simulation–optimization model for solving flexible flow shop scheduling problems with rework and transportation

Elmira Gheisariha[a], Madjid Tavana[b,c,*,1], Fariborz Jolai[d], Meysam Rabiee[e]

[a] *Department of Industrial Engineering, Faculty of Industrial and Mechanical Engineering, Qazvin Islamic Azad University (QIAU), Qazvin, Iran*
[b] *Business Systems and Analytics Department, Distinguished Chair of Business Analytics, La Salle University, Philadelphia, PA 19141, USA*
[c] *Business Information Systems Department, Faculty of Business Administration and Economics, University of Paderborn, D-33098 Paderborn, Germany*
[d] *Department of Industrial Engineering, University of Tehran, Tehran, Iran*
[e] *Lundquist College of Business, University of Oregon, Eugene, USA*

Received 15 August 2019; received in revised form 14 June 2020; accepted 19 August 2020
Available online 1 September 2020

Abstract

We propose an enhanced multi-objective harmony search (EMOHS) algorithm and a Gaussian mutation to solve the flexible flow shop scheduling problems with sequence-based setup time, transportation time, and probable rework. A constructive heuristic is used to generate the initial solution, and clustering is applied to improve the solution. The proposed algorithm uses response surface methodology to minimize both maximum completion time and mean tardiness, concurrently. We evaluate the efficacy of the proposed algorithm using computational experiments based on five measures of diversity metric, simultaneous rate of achievement for two objectives, mean ideal distance, quality metric, and coverage. The experimental results demonstrate the effectiveness of the proposed EMOHS compared with the existing algorithms for solving multi-objective problems.

Keywords: Flexible flow shop scheduling; multi-objective harmony search; Gaussian mutation; Simulation and computational experiments; Sequence-dependent setup times; Response surface methodology

1. Introduction

Developing efficient scheduling systems is an essential factor for the productivity of the manufacturing systems in today's competitive marketplace. The flexible flow line environment is a relatively common scheduling system appearing in the flow industries such as steel, petroleum, chemical processing, and packaging [1]. However, the current scheduling models do not consider the transportation time, probable rework, and sequence-dependent setup times in multi-objective scheduling problems. The classical flow shop problems involve only one machine at each stage. More than one machine at least at each stage turns the problem into a flexible flow-shop problem.

* Corresponding author at: Business Systems and Analytics Department, Distinguished Chair of Business Analytics, La Salle University, Philadelphia, PA 19141, United States.
E-mail addresses: e.gheisariha@alumni.ut.ac.ir (E. Gheisariha), tavana@lasalle.edu (M. Tavana), fjolai@ut.ac.ir (F. Jolai), meysamr@uoregon.edu (M. Rabiee).
[1] http://tavana.us/.

https://doi.org/10.1016/j.matcom.2020.08.019

Hence, the flexible flow shops are the generalized forms of simple flow shops. Chang and Liao [8] called the scheduling jobs in the flexible flow shops NP-complete problems. The focus of most research in the flow shop literature has been on the single-objective optimization. Kurz and Askin [34] proposed an integer-programming model for a hybrid flow shop (HFS) with sequence-dependent setup times (SDST), which is one of the studies that use a meta-heuristic approach. They developed a random key genetic algorithm to solve the problem against other heuristic rules due to the difficulty in developing a direct solution to the integer programming model. The authors introduced a genetic algorithm to resolve the problem. Lin et al. [37] provided a simulated annealing-based meta-heuristic for reducing the make-span in a flow-shop manufacturing cell with sequence-dependent family setup times. The algorithm proposed was compared in terms of effectiveness and efficiency with available heuristics on a benchmark problem dataset previously used in other studies. An efficient hybrid metaheuristic for scheduling jobs was presented by Behnamian et al. [4] in a hybrid flow-shop with sequence-dependent setup times. Their goal was to develop a schedule capable of minimizing the sum of the earliness and tardiness of jobs. Tardiness has a direct impact on customer satisfaction and profitability. In addition, there is an inverse relationship between makespan and throughput since minimizing makespan results in maximizing throughput. Naderi et al. [48] proposed hybrid flow shops where the setup times are sequence-dependent to minimize makespan and maximum tardiness. They introduced a novel simulated annealing with a migration mechanism and compared their algorithm with several high-performing metaheuristics. Readers can refer to Wang [63] for a comprehensive review of the scheduling problems with a flexible flow shop.

Recent studies have focused on the problem inherent in the single-machine scheduling with sequence-dependent setup times aimed at minimizing the total weighted tardiness of jobs. Chen [9] presented an iterated population-based variable neighborhood descent search algorithm designed to address a single machine with sequence-dependent setup times. This algorithm provided initial population solutions followed by the neighborhood descent search algorithm generated for each iteration. Subsequently, the solutions were enhanced by a greedy local search. Cheng et al. [12] developed a mixed-integer linear programming model for a no-wait flow shop scheduling problem with SDST and proposed a pairwise iterated greedy algorithm to solve the large size problems. Da Silva et al. [14] studied an online single machine scheduling problem with SDST. They proposed a mixed-integer linear programming formulation for the problem along with some heuristic algorithms. [31] proposed three metaheuristic algorithms (hybrid squirrel search algorithm, opposition based whale optimization algorithm, and discrete gray wolf optimization algorithm) to solve bi-criteria SDST hybrid flow shop scheduling problems. Appendix A presents a comprehensive list of all acronyms and abbreviations used in this paper.

Naderi et al. [47] reviewed the drawbacks of available models in the literature. Their work led to the development of four mixed-integer linear programming models, which were compared with each other in terms of size and computational complexity. Additionally, they proposed a novel hybrid particle swarm optimization algorithm. The research by Ribas et al. [56] and Ruiz and Vázquez-Rodríguez [57] can be used for further information and a better understanding of the hybrid flow shop scheduling problem. Hwang and Lin [27] proposed a two-stage flexible flow shop with four standard performance measures, namely, total completion time, maximum lateness, total tardiness, and the number of tardy jobs. They specified an optimum interleaving processing sequence of all the jobs associated with their starting times on the stage-2 bottleneck machine. Sioud and Gagné [59] proposed an enhanced migrating birds optimization algorithm to solve the flow shop problem with sequence-dependent setup times. They aimed to minimize the makespan using a novel approach based on the setup times and the new enhanced migrating birds optimization algorithm, namely a new heuristic based on setup times for the permutation flow shop with SDST, and an enhanced migrating birds optimization algorithm, respectively. Zhang et al. [69] proposed a multi-objective optimization model to solve the hybrid flow shop scheduling problems by minimizing total energy consumption and makespan. They proposed a decomposition-based metaheuristic algorithm for solving large-scale problems. Fernandez-Viagas et al. [21] examined the efficiency of different solution representations for solving hybrid flow shop scheduling problems. They found a trade-off between the ability to conduct an efficient search in this reduced solution space and the solution space reduction. Bargaoui and Driss [3] proposed a multi-stage model using the tabu search for solving the permutation flow shop scheduling problems. They used two multi-agent classes of supervisor and scheduler agents. The supervisor agent generates the initial solution with the tabu search core, and the scheduler agents are responsible for the evaluation of all neighborhood solutions and satisfaction of the constraints. Computational experiments on different benchmarks showed that their proposed model reaches a high-quality solution. The model proposed in this study is devised to minimize the makespan or the total duration of the schedule.

There have also been numerous studies in the scheduling literature focusing on multi-objective and hybrid flow shop scheduling problems. However, most of them have merely followed a single dimension pattern, i.e., either single objective hybrid flow shops, or a multi-objective optimization in scheduling such as single-machine or classical flow shops. Another study by Tran and Ng [62] resulted in a hybrid water flow algorithm for multi-objective flexible flow shop scheduling problems. It was designed to minimize the completion time of jobs and the total tardiness time of the jobs. According to Meng et al. [45], during the optimization, the lot-splitting is suitable in advance and fixed in the flexible flow shop problem. They used the lot-streaming flow shop scheduling problems and an improved migrating birds optimization for minimizing the maximum completion time or the make-span. They also presented a mathematical model to solve the *n*-job *m*-machine lot-streaming flow shop scheduling problem with separable sequence-independent setup times and equal-size sub-lots. A hybrid metaheuristic algorithm was proposed by Behnamian and Ghomi [5] designed to address a hybrid flow shop scheduling with the machine and resource-dependent processing times. The result of Yagmahan and Yenisey [65] was a multi-objective ant colony system algorithm, featured with combining the ant colony optimization approach and a local search strategy to resolve this scheduling problem. This was the first use of an ant colony optimization metaheuristic regarding multi-objective m-machine flow shop scheduling problems aimed at both objectives of make-span and total flow time. A hybrid multi-objective backtracking search algorithm to solve an energy-efficient permutation flow shop scheduling problem with controllable transportation time was developed by Lu et al. [38]. Behnamian et al. [6] described a multi-phase algorithm covering the Pareto optimal front hybrid metaheuristic to minimize the make-span and the sum of the earliness and tardiness of jobs. Yagmahan and Yenisey [64] investigated the flow shop scheduling problem with a focus on minimizing the make-span, total flow time, and total machine idle time. The ant colony optimization algorithm was employed to solve this problem. A limited monkey algorithm proposed by Marichelvam et al. [42] was used to solve the flow shop scheduling problem. It was demonstrated that the monkey algorithm is useful due to its simple structure and strong robustness. Recently, Nouri et al. [52] studied flexible job shop scheduling problems with transportation times and a single robot. They employed a new metaheuristic hybridization approach based on clustered holonic multi-agent models, and they used a local search with a set of cluster agents to improve the quality of the final population. Nouri et al. [51] presented a neighborhood-based genetic algorithm for a flexible job-shop scheduling problem with transportation time and multiple robots. They introduced a new metaheuristic hybridization approach based on a clustered holonic multiagent model. Nouri et al. [49] reviewed the job shop scheduling problems with transportation resources according to seven criteria, including transportation resource number, transportation resource type, job complexity, routing flexibility, recirculation constraint, optimization criteria, and the implemented approaches. Nouri et al. [50] proposed another interesting model for the simultaneous scheduling of jobs and robots in job shop scheduling.

Li et al. [36] presented a multi-stage HFS with single and batch processing machines to minimize the maximum completion time and the total weighted tardiness by developing a heuristic-search genetic algorithm. More recently, Dios et al. [17] compared the hybrid flow shop scheduling with missing operations (HFSMO) with the traditional HFS problems. The proposed algorithm turned out to be more effective in solving the HFSMO problem than the existing heuristics.

A Lorenz dominance based on the non-dominated sorting genetic algorithm (NSGAII) was presented by Dugardin et al. [18] to address a reentrant hybrid flow shop scheduling problem by maximizing the utilization rate of the bottleneck and minimizing the maximum completion time. According to the results, the Lorenz NSGAII can provide better solutions than the NSGAII and the strength Pareto evolutionary algorithm 2. Ebrahimi et al. [19] suggested a scheduling problem in an HFS environment, using a sequence-dependent family setup time to minimize the make-span and total tardiness. They used a normal data distribution and assumed uncertain due date in their model. They presented two metaheuristic algorithms (NSGAII and multi-objective genetic algorithm) and compared the quantitative and qualitative results of these two algorithms with a multi-phase genetic algorithm. Gholami-Zanjani et al. [24] considered a flow shop scheduling problem with sequence-dependent set-up times using robust and fuzzy optimization in an uncertain environment aimed at minimizing the weighted mean completion time. Considering a branch-and-bound algorithm for two-machine flow-shop scheduling problems with batch delivery costs, Mazdeh and Rostami [43] managed to minimize the maximum tardiness plus the sum of delivery costs. A mixed-integer linear programming model and a branch and bound algorithm were also developed to solve the problem.

Marichelvam and Geetha [41] used a harmony search algorithm with a real-industrial scheduling problem and proposed and analyzed a multistage HFS scheduling problem. Jabbarizadeh et al. [28] developed hybrid flexible flow

shops with sequence-dependent setup times and machine availability constraints caused by preventive maintenance. The optimization criterion included the minimization of the make-span. Sokolov et al. [61] focused on a flexible flow shop problem for continuous production. Using uniform alternative machines, they analyzed the theories of the optimal scheduling approach. The proposed method can be used in the mathematics of functional spaces, including stability, robustness, controllability, adaptability. A multi-phase genetic algorithm developed by Karimi et al. [29] addresses the bi-objective group scheduling in a hybrid flexible flow shop. By incorporating flexible and diverse maintenance activities to minimize the total tardiness and maintenance costs, a permutation flow shop scheduling problem was proposed by Yu and Seif [66] as a mixed-integer linear program. A lower-bound-based genetic algorithm was presented that initially examines the parameters using a factorial experiment to identify the statistically significant ones. Zandieh and Karimi [68] employed a multi-objective group scheduling problem in a hybrid flexible flow-shop with sequence-dependent setup times by simultaneously minimizing the total weighted tardiness and maximum completion time. A multi-population genetic algorithm was proposed by them to search for a Pareto optimal solution, which was compared with two well-established benchmarks, a multi-objective genetic algorithm and NSGAII using three sizes of test problems, including small, medium and large to evaluate the performance of the proposed multi-population genetic algorithm.

Considering the literature reviewed above and to the best of our knowledge, the flexible flow shop scheduling problem with SDST, transportation time by conveyor, ready time, and probable rework have not been studied with two concurrent objectives. Thus, this research appears to be the first attempt in applying a new multi-objective algorithm called the novel harmony search algorithm with the Gaussian mutation. We chose five measures to assess the performance of our proposed algorithms. We used the performance measures to compare the results of the multi-objective algorithms quantitatively, followed by the normalization of both objective functions. The rest of the paper was organized as follows. In Section 2, the problem description was provided, and the assumptions were introduced in detail in Section 3. The multi-objective novel harmony search algorithm with the Gaussian mutation, generating an initial solution with the MCH algorithm, and the data-mining approach was described in Section 4. Section 5 focused on how to generate the test data, parameter tuning, and analyses of computational experiments. Finally, in Section 6, we presented the conclusions and some promising directions for future research in the area.

2. Problem definition

The flexible flow shop is an example of a machine scheduling problem. There are n jobs to be processed. Job j, $(j = 1, 2, \ldots, n)$ is processed at each stage i, $(i = 1, 2, \ldots, s)$ in the series. There are m_i identical parallel machines at the stage i. The processing times of job j at the ith stage is represented by p_{ij}. The sequence-dependent setup time between job j and job k at the ith stage is depicted by s^i_{jk}. For the transportation of the job j between stage i and l, we define three terms as loading time, traveling time, and unloading time, which are denoted by lt^j_{il}, tt_{il} and ut^j_{il}, respectively. After processing each job at each stage on each machine, an inspection procedure was considered. The inspection time is a segment of the processing time. After inspection, with predetermined probability (rp^i_j), each job may be needed for the reworking procedure defined by the rework time (rt^i_j). Fig. 1 illustrates the problem.

The problem was divided into three sub-problems: finding the best sequence for jobs, assignment of jobs to machines for processing by using the first available machine [25,53] and designing a strategy for jobs in the transportation step employing the first-in-first-out and sequence priority. We conducted some preliminary runs to select the rule where the sequence priority rule was chosen due to better performance compared to first-in-first-out based on the following assumptions: We conducted some preliminary runs to select a sequence priority rule that performed better than the first-in-first-out rule, based upon the following:

- No preemption,
- Processing each job once,
- Processing times are independent of the schedule and different at different stages,
- No machine can process more than one job simultaneously,
- Machines may be idle, waiting for the next job,
- Machines never breakdown,
- The system has no buffers between stages,
- All jobs go through all production stages in the same order,

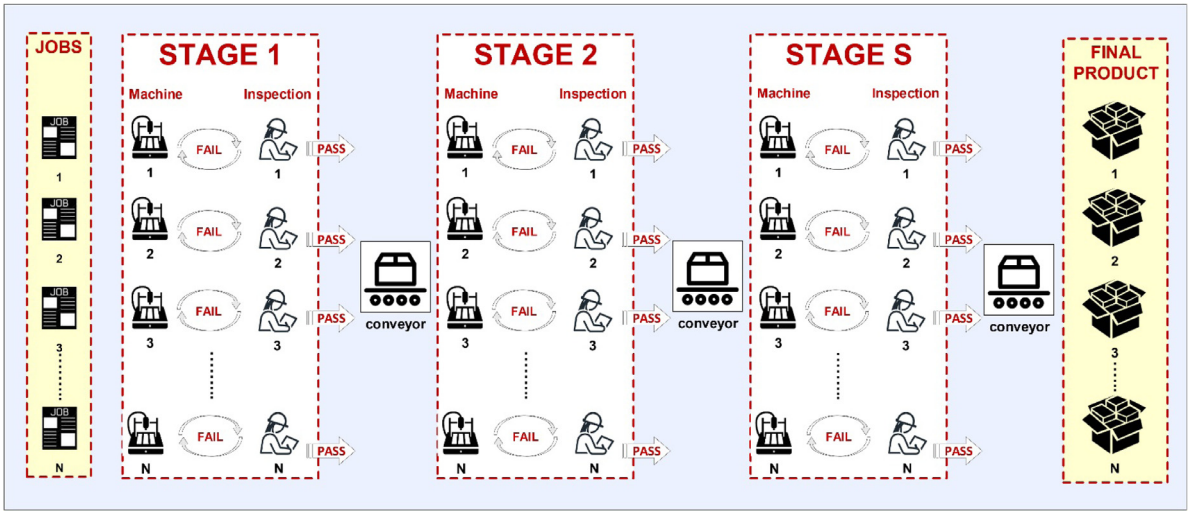


Fig. 1. A graphical representation of the problem.

- Processing time is equal in all processors of each stage,
- Each stage contains at least one processor. With more than one processor, they are identical, and
- Transportation time is job independent and is related to the distance between two consecutive stages.

3. Mathematical formulation

The following notations, variables, and an optimization model are used to formulate the flexible flow shop scheduling problem.

Notations:

n	The number of jobs
s	The number of stages
j, k	Index for jobs $\{1, 2, \ldots, n\}$
i, l	Index for stages $\{1, 2, \ldots s\}$
b	Index for process or rework $\{2, 3, \ldots, Q\}$, where Q is a big number.
t	Index for machine $\{1, 2, \ldots s\}$
S_j	The last stage for job j
M_i	The number of machines at stage i
P_{ij}	The processing time of job j at the stage i
P_{ijb}	The processing time of job j at stage i related to the process or rework
P_{ri}^{j}	The probability of reworking job j at the stage i
d_j	The due date of the job j
S_{jk}^{i}	The sequence-dependent setup time of job j, k at the stage i
lt_{tl}^{j}	The load time of job j at the stage $l$on the machine t
tt_{il}^{j}	The transportation time between$i, l, i = \{1, 2, \ldots, n-1\}, p = \{2, 3, \ldots, n\}$
ut_{tl}^{j}	The unload time of job j at the stage i on the machine t

Variables:

C_j^i	The completion time of job j at the stage i
C_j	The completion time of job j, which is equal to $Max(C_j^i) \forall i = 1, \ldots, s$
T_j	The tardiness of job j

$$x_{jk}^{i} = \begin{cases} 1 & \text{if Job } j \text{ is scheduled afte Job } k \text{ in the } s\text{tage } i \\ 0 & \text{otherwise} \end{cases}$$

Maximizing production line utilization and minimizing work-in-progress inventories are two common concerns in manufacturing systems. The following two common objectives are used to address these two concerns:

- Minimization of maximum completion time or makespan (min ($C_{\max}$))
- Minimization of average completion time or total completion time (min ($\sum C_j$))

Optimization model:

$$Min \;\; z = (Z_1, Z_2) \tag{1}$$

s.t.

$$\sum_{j=1}^{n} x_{0j}^{i} = m_i, i = 1, 2, \ldots, s \tag{2}$$

$$\sum_{k \in \{s^l, n+1\}} x_{jk}^{i} = 1, j = 1, \ldots, n, j \in e_j \tag{3}$$

$$\sum_{j \in (0, s^l)} x_{jk}^{i} = 1, k = 1, \ldots, n, i \in e_j \tag{4}$$

$$C_j^i - C_k^i + M^i(1 - x_{jk}^i) \geq P_{ijb} + S_{jk}^i, k = 0, \ldots n, j = 1, \ldots, n \tag{5}$$

$$C_j^i - C_j^{i-1} + M_j^i(1 - x_{jk}^i) \geq P_{ijb} + S_{jk}^i, k = 0, \ldots, n, j = 1, \ldots, n, i \in e_j - \{1\}, b = 2, \ldots, Q \tag{6}$$

$$C_j^i \geq C_0^i, j = 1, \ldots, n, i = 1, \ldots, s \tag{7}$$

$$P_{ijb} = P_{ij} + P_{ij}(b-1)P_{ri}^j, i = 1, \ldots, s, j = 1, \ldots, n, b = 2, \ldots, Q \tag{8}$$

$$C_j^{l+1} + ut_{tl}^j \geq C_j^l + tt_{l,l+1}^j + lt_{tl}^j \tag{9}$$

$$Z_1 \geq C_j^{sj}, j = 1, 2, \ldots, n \tag{10}$$

$$T_j \geq C_j^{sj} - d_j, j = 1, \ldots, n \tag{11}$$

$$T_j \geq 0, j = 1, \ldots, n \tag{12}$$

$$Z_2 = 1/n \sum_{j=1}^{n} T_j, j = 1, \ldots, n \tag{13}$$

$$\begin{cases} x_{jk}^i \in \{0, 1\}, j, k = \{1, \ldots, n, n+1\}, i = 1, \ldots, s \\ x_{jk}^i = 0, j = k, i = 1, \ldots, s \\ C_j^i \geq 0, j = 0, 1, \ldots, n, i = 1, 2, \ldots, s \end{cases} \tag{14}$$

Constraint 1 calculates min Z. $Z_1 = \max(C_j^{sl})$, and $Z_2 = T_j = Max(0, C_j - d_j)$. Constraint 2 indicates that the m_i machines are timed at each stage. t_{0j}^i reveals that the first job processed in every i Stage in Job j. Constraints 3 and 4 guarantee the performing timing of each job on only a single machine at every stage. Constraint 5 guarantees that Job k is timed right after Job j is timed with the least waiting time. Also, Constraint 6 indicates that the time spent on performing Job j at Stage i equals its processing time at Stage $i-1$ plus its processing time at Stage i. m_j^i is computed by $m_j^i = P_{i,j}$. Constraints 5 and 6 suggest that no preparation time can be considered for Job j unless it can be accessed at Stage i. Constraint 7 guarantees that the completion time of a job which has not yet been assigned to Stage i equals its completion time at Stage $i-1$. Constraint 8 shows the possibility of reworking Job j at Stage i. Constraint 9 guarantees that the completion time of Job$j$ at Stage $l+1$ plus the collection time of job j from Machine t at Stage l is greater than the completion time of Job j at Stage l plus the transportation time from Stage l to Stage $l+1$ and the uploading of Job j onto Machine$t$ at Stage l. Constraint 10 shows the relationship between C_j^{sj} and Z_1. Constraints 11 and 12 show the amount of delay and its positive value, respectively. Constraint 13 indicates the relationship between the total delay and Z_2. Constraint 14 indicates the status of the variables.

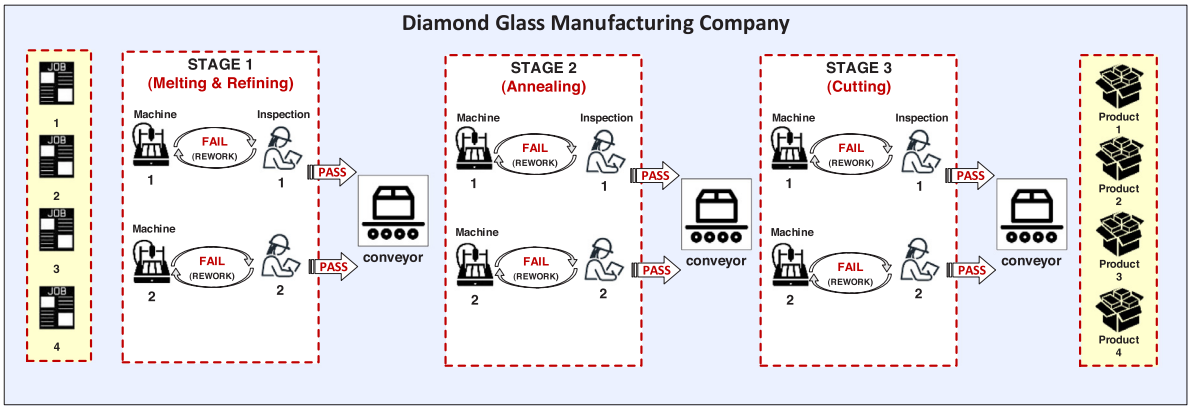


Fig. 2. Problem scenario at Diamond Glass Manufacturing Company.

Table 1
Processing time for each job in three stages.

i/j	1	2	3	4
1	3	3	8	5
2	7	4	2	6
3	5	7	6	2

3.1. Problem scenario

In this section, we present a case problem at Diamond Glass Manufacturing Company[2] to demonstrate the applicability of the model proposed in this study. As shown in Fig. 2, the case involves processing four jobs. Each job is processed in three stages of melting and refining, annealing, and cutting. In the melting and refining stage, fine-grained ingredients are controlled for quality and mixed to make a batch flown into the furnace heated up to 1500 °C for melting. In the annealing stage, the glass moves through the annealing lehr to remove internal stress and make the glass more prone to cutting. Finally, in the cutting stage, the diamond steels trim-off selvage, stressed edges, and cut the ribbon to the job size. Two machines are available to complete the necessary work at each stage. Sequence-dependent preparation (i.e., cleaning, tooling, etc.) is needed to get each machine ready for processing the job at each stage. Once a job is processed on a machine, the product is inspected. If the product passes the inspection, it moves to the next stage. Otherwise, the product goes through a secondary process (i.e., rework option). Workers operate the conveyors between stages (loading and unloading) to cool or heat the product, depending on the job.

As described above, there are four jobs, three stages, and two machines at each stage. Therefore, $N = 4$, $M = 2$, and $S = 4$ with the release times and due dates given as follows:

$$r_1 = 0, r_2 = 0, r_3 = 0, and\ r_4 = 0; d_1 = 54, d_2 = 45, d_3 = 40, and\ d_4 = 26$$

The processing times of jobs at each stage are given in Table 1, setup times in Table 2, and transportation times in Table 3. The probability of rework and rework time are shown in Table 4.

Four random numbers were generated between 0 and 1 (0.34, 0.78, 0.12, 0.24) for four jobs (3, 4, 1, 2) where the smallest random number is associated with Job 1, and the largest random number is associated with Job 4. The remaining jobs are placed in between.

The machines were selected according to the first available machine. Job sequencing was determined based on the processing time, setup time, and transportation time (Fig. 3). For example, a random number between 0 and 1 was generated for Job 3, and it was compared to the rework probability. If the value were smaller than the rework

[2] The name is changed to protect the anonymity of the company.

Table 2
Sequence-dependent setup times for each job at three stages.

k/j	Stage 1				Stage 2				Stage 3			
	1	2	3	4	1	2	3	4	1	2	3	4
1	2	2	1	4	1	2	4	5	1	6	4	2
2	5	4	6	2	3	3	4	2	2	2	1	5
3	6	2	1	4	5	4	2	3	1	3	3	6
4	2	4	1	2	2	1	3	2	4	4	3	1

Table 3
Load, transportation, and unload times.

(a) Between Stage 1 and Stage 2

Time	Job			
	1	2	3	4
Load time	3	4	4	2
Transportation time	2	4	3	5
Unload time	3	4	5	3

(b) Between Stage 2 and Stage 3

Time	Job			
	1	2	3	4
Load time	1	4	3	2
Transportation time	3	2	1	1
Unload time	1	2	3	1

Table 4
Probability of rework time and rework time for each job at three stages.

(a) Probability of rework time

i/j	Jobs			
	1	2	3	4
1	$\frac{3}{100}$	$\frac{11}{100}$	$\frac{4}{100}$	$\frac{5}{100}$
2	$\frac{6}{100}$	$\frac{7}{100}$	$\frac{3}{100}$	$\frac{7}{100}$
3	$\frac{8}{100}$	$\frac{10}{100}$	$\frac{5}{100}$	$\frac{7}{100}$

(c) Rework time

i/j	Jobs			
	1	2	3	4
1	2	2	3	3
2	4	2	1	3
3	2	3	4	1

probability, the rework process would be applied; otherwise, it was the same as in the previous case. In stage 3 for job 3, the random number of 0.04 was obtained, which was smaller than 0.05, and the reworking was applied to job 3. The time for this reworking was 0.04. Consequently, for job 1 in stage 1 and job 2 in stage 2, the reworking was applied. According to Fig. 3, $C_{\max}$ is defined as:

$$C_{\max} = \max(36, 50, 40, 29) = 50, C_j = (36, 50, 40, 29), T_j = [(54-36), (45-50), (40-40), (26-29)]$$
$$= (5, 3), \ \overline{T} = \frac{5+3}{4} = 2 \rightarrow C_{\max} = 50, \overline{T} = 2.$$

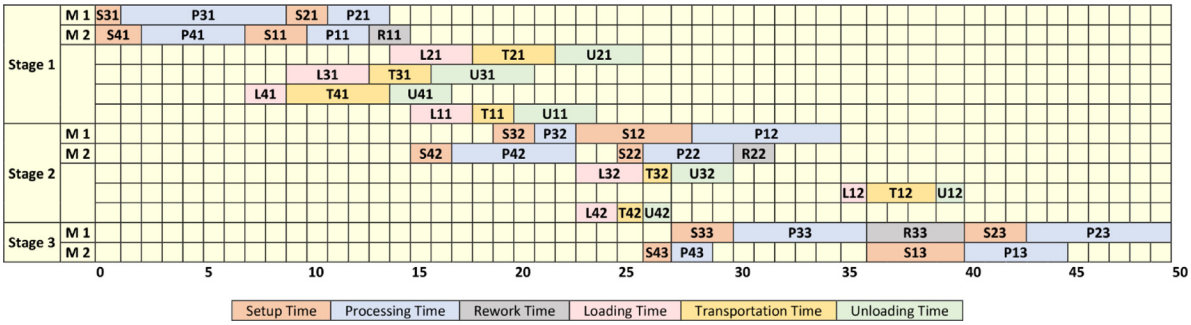


Fig. 3. The job allocation procedure for the illustrated example.

Table 5
Processing time for each job at two stages.

i/j	Jobs		
	1	2	3
1	3	3	8
2	7	4	2

Table 6
Sequence-dependent setup times for each job at two stages.

k/j	Stage 1			Stage 2		
	1	2	3	1	2	3
1	2	2	1	1	2	4
2	5	4	6	3	3	4
3	6	2	1	5	4	2

3.2. The lower bound (LB)

In this section, we propose a lower bound to validate our results for the objective function. We assume all machines are always available for processing jobs to simplify the calculation procedure. In other words, we assume there is no waiting time for the jobs (no idle time for the machines). We further disregard the rework option since it is a probabilistic event that cannot be incorporated in the lower bound calculation. Therefore, the lower bound proposed here is applicable to deterministic cases similar to the relaxed version of the problem considered in this study. The formula for calculating the lower bound is presented as Eq. (15):

$$\mathrm{LB} = Max\left\{\sum_{i=1}^{S} P_{ij} + Min\, S_{jk}^{1} + \sum_{i=1}^{S}\left(lt_{il}^{j} + tt_{il}^{j} + ut_{il}^{j}\right)\right\} \forall k, j \in 1,\ldots,n,\ l \in 1,\ldots,S \tag{15}$$

where P_{ij} is the processing time of job j in Stage i, $MinS_{jk}^{1}$ is the minimum sequence-dependent setup time for each job in Stage 1. It is worth noting that the sequence-dependent setup times of the other stages are disregarded since we assume all machines are available right after transporting jobs to the next stage. lt_{il}^{j}, tt_{il}^{j} and ut_{il}^{j} are the loading, transportation, and unloading times for each job between two subsequent stages, respectively.

Let us consider the following numerical example with three jobs to show the calculation procedure. This is a two-stage flexible flow shop problem with two identical machines at each stage. There are 3! possible sequence for these three jobs as follows: 1-2-3, 1-3-2, 2-1-3, 2-3-1, 3-1-2 and 3-2-1. The inputs for this numerical example are shown in Tables 5, 6, and 7. Table 5 shows the processing time of the three jobs in Stage 1 and Stage 2. Table 6 shows the sequence-dependent setup time for jobs in both stages, and finally, the loading time, transportation time, and unloading time are shown in Table 7.

Table 7
Load, transportation, and unload times between Stage 1 and Stage 2.

Time	Jobs		
	1	2	3
Load time	3	4	4
Transportation time	2	4	3
Unload time	3	4	5

The following optimal $C_{\max}$ is calculated for the six possible sequences:

$$Min\, C_{\max} = Min(28, 26, 28, 26, 33, 33) = 26$$

Appendix B shows the maximum completion time for all six job sequences. Next, we calculate $C_{\max}$ for job sequences composed of three jobs, two stages, and two machines. According to Eq. (15), a lower bound of 23 is calculated for this problem as follows:

$$\text{LB} = Max\,\{(10, 7, 10) + (2, 2, 1) + (8, 12, 12)\} = Max\,\{(20, 21, 23)\} = 23$$

The closeness of 23 to 26 derived from the lower bound formula proposed here, confirms and validates our results.

4. Enhanced multi-objective harmony search algorithm with Gaussian mutation (E-MOHS)

The meta-heuristics have been successfully applied to a variety of optimization problems such as short-term hydro system scheduling [67], flexible job-shop scheduling problems [58], cost minimization of the butter-oil processing plant [55], and global optimization [39]. Harmony search is one of the widely used metaheuristic algorithms that is inspired by musical improvisation. There more advantages to the harmony search algorithm. The main advantage of harmony search is in its ability to control a few decision variables, not requiring initial value settings, not depending on differentiability and continuity as a mathematical characteristic, and ease of implementing and understanding [32]. In addition, the harmony search algorithm has shown poor convergence and poor global search. Consequently, the algorithm memory-based constructive heuristics is used to generate the initial solutions, and the proposed data mining method in the algorithm is used for improving convergence. It has been employed for solving different optimization problems [2,44,70]. Dai et al. [15] proposed the multi-objective version of this algorithm, known as MOHS, and several important enhancements are included in the proposed harmony search algorithm with Gaussian mutation (GMHS). First, each harmony stored in the harmony memory is selected with different probabilities to improve the convergence of the harmony search algorithm. Second, in the pitch adjustment phase, a decreasing bandwidth plus an adaptive bandwidth are designed to improve both the global and local search ability throughout the search process. The third is the integration of chaotic maps with GMHS for parameter adaption. Finally, a Gaussian mutation operator is utilized after improvisation in the GMHS to speed up the convergence rate and to prevent the trapping of the algorithm in local optima.

4.1. The basic harmony search algorithm

We called each solution in the basic harmony search a "harmony" and described it by an n-dimension real vector, which was introduced by Geem et al. [23]. In the beginning, an initial population of harmony vectors is randomly generated and stored in a harmony memory (hm). In the case of the better fitness value of the newly generated harmony than the worst one in the hm, the hm will be updated. In the following, the basic harmony search algorithm procedure is presented:

Step 1: Initializing the problem and harmony search parameters

In this stage, the parameters of the harmony search algorithm are set: Harmony memory size, harmony memory considering rate (HMCR), distance bandwidth (bw), pitch adjusting rate (PAR), and the number of improvisations.

Step 2: Initializing the harmony memory

$$\mathrm{hm} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_i \\ \vdots \\ X_{\mathrm{HMS}} \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,j} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,j} & \cdots & x_{2,n} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ x_{i,1} & x_{i,2} & \cdots & x_{i,j} & \cdots & x_{i,n} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ x_{\mathrm{HMS},1} & x_{\mathrm{HMS},2} & \cdots & x_{\mathrm{HMS},j} & \cdots & x_{\mathrm{HMS},n} \end{bmatrix} \tag{16}$$

Step 3: Innovating a new harmony

Based on the three rules of (a) memory consideration, (b) pitch adjustment, and (c) random selection, we produced a new harmony vector of $X_{new} = (X_{new,1}, X_{new,2}, \ldots, X_{new,n})$. The generation of harmony is typically called "Improvisation", which was proposed by Lee and Geem [35]. In the memory consideration phase, we randomly selected the value of each decision variable $X_{new,j}\,(j \in \{1, 2, \ldots, n\})$ for the new harmony from any harmony vector $X_{i,j}\,(i \in 1, 2, \ldots, \mathrm{HMS})$ with the probability of HMCR.

$$x_{new,j} \leftarrow \begin{cases} x_{new},\ j \in \left[x_{1,j}, x_{2,j}, \ldots., x_{\mathrm{HMS},j}\right]; \\ \quad with\ probability\ \mathrm{HMCR} \\ x_{new,j} = \mathrm{LB}_j + rand\,() \,.\, \left(\mathrm{UB}_j - \mathrm{LB}_j\right); \\ \quad with\ probability\ 1 - \mathrm{HMCR}, \end{cases} \tag{17}$$

Using the pitch adjustment rule with the probability of PAR, we set each $X_{new,j}$ achieved by the memory consideration so that the proportion of $1 - \mathrm{PAR}$ and the value of $X_{new,j}$ would remain unchanged. Under the pitch adjustment rule, $X_{new,j}$ was set as follows:

$$x_{new,j} = \begin{cases} x_{new,j} \pm rand\,() \cdot bw; & with\ probability\ \mathrm{PAR} \\ x_{new,j}; & with\ probability\ 1 - \mathrm{PAR} \end{cases} \tag{18}$$

The memory-based constructive heuristic is used to generate an initial solution for the harmony search algorithm. We use the MCH-12 algorithm [20] to obtain the lower-bound solutions to the problem (See Algorithm 1 in Appendix C for the MCH-12 pseudo-code). A novel clustering approach is applied to the population in the harmony search memory in each iteration to improve the quality and diversity of obtained solutions in the Pareto front. The clustering approach enhances the exploration and exploitation of our proposed algorithm. The clustering structure is originated from the k-means method with a subtle difference. The user defines the number of clusters (k) as an input at the beginning of the algorithm. The following mathematical function is used for this purpose:

$$Total\ WCV = \sum_{i=1}^{n} (E_i - x_i^*)^2 \tag{19}$$

where k is the total number of clusters and E_i is the value of the ith element in each solution, x_i^* is the solution representation (i.e., permutation) of the optimal solution, and n is nth location in the solution representation. In the k-means method, μ is replaced with x_i^* in our proposed method.

This method calculates the distance from the optimal solution for all solutions in the pool. They are then sorted based on their distance in descending order. Next, each solution is assigned to a predefined k cluster from the nearest to the longest distance. Those with smaller distances from the optimal solution (the best cluster) will receive a higher probability, and those with a longer distance from the optimal solution (the worst cluster) will receive a lower probability. Finally, the MOHS will consider these probabilities in its exploration and exploitation procedures. Since our problem is a multi-objective problem, this process is repeated twice in each iteration—one time based on $C_{\max}$ and one time for the mean tardiness.

Step 4: Updating the harmony memory

A better newly generated harmony X_{new} than the worst harmony stored in the hm assessed by the objective function value leads to updating the hm. In this case, we replaced the worst harmony with a new one.

Step 5: Evaluating the stop measure

If the stop measure is met, running the algorithm will be terminated. Otherwise, we continue to phase 3.

4.2. The suggested GMHS for the multi-objective optimization

Using the basic harmony search algorithm for solving multi-objective optimization problems (MOOPs), a weak function was revealed by the algorithm in escaping from the local Pareto front as well as a poor convergence rate. Thus, we suggested a GMHS solve the MOOPs.

4.2.1. Modified memory consideration

Chen et al. [10,11] proposed a small value in the case of uni-objective optimization. As the ith harmony vector was generated, its analogous harmony in hm would be assigned in the ith position of hm. In a modified memory consideration, the value of every decision variable $X_{new,j}$for the new harmony is randomly produced within the possible range of values, with a probability $1-$ HMCR, while choosing the probability of HMCR, $X_{new,j}$ is from any harmony vector $X_{i,j}$in the hm with varied probabilities instead of the identical probability.

$$x_{new,j} \leftarrow \begin{cases} x'_{new,j}; \\ \quad with\ probability\ \text{HMCR} \\ x_{new,j} = \text{LB}_j + rand\,() \cdot \left(\text{UB}_j - \text{LB}_j\right); \\ with\ probability\ 1 - \text{HMCR} \end{cases} \tag{20}$$

$$x'_{new,j} \leftarrow \begin{cases} x'_{new,j} \in \{x_1, j, x_2, j, \ldots, x_{\text{HMS}}, j\} \\ with\ probability\ P_C \\ x_{corresponding,j} \\ with\ probability\ 1 - P_C \end{cases} \tag{21}$$

According to the subscript, 'new' of$X_{new,j}$, the $X_{new,j}$ is a newly produced harmony, $new \in \{1, 2, \ldots, \text{HMS}\}$. In Eq. (21), the parameter P_c occurs in the range of [0, 1]. and the probability of all elements in the ith harmony is stored in the hm chosen to generate the newth harmony as Eq. (22), but without P_c, this probability can be formulated as Eq. (23). P_i^{HS} is not dependent on P_c, while P_i^{GMHS} does. Also, P_i^{GMHS} is almost equivalent to P_i^{HS} while P_c has a value close to 1 and $new \neq i$, the P_i^{GMHS} is always much bigger than P_i^{HS} when $new = i$.

$$P_i^{\text{GMHS}} = \begin{cases} \left(1 - P_c + \dfrac{P}{\text{HMS}}\right)^n; & new = i \\ \left(\dfrac{P_c}{\text{HMS}}\right)^n; & otherwise \end{cases} \tag{22}$$

$$P_i^{\text{HS}} = \left(\frac{1}{\text{HMS}}\right)^n, new = 1, 2, \ldots., \text{HMS} \tag{23}$$

4.2.2. Chaotic map-based parameter adaption for PAR

Due to the dependence of parameter PAR on the problem features, an adaptive PAR is preferred. Two chaotic maps were used in the pitch adjustment phase of GMHS for parameter matching as follows:

Iterative chaotic map with infinite collapses (ICMIC) map: the following equation defines the iterative chaotic map with infinite collapses proposed by He et al. [26].

$$x_{n+1} = \sin\left(\frac{a}{x_n}\right), a \in (0, \infty), x_n \in (-1, 1) \tag{24}$$

The ICMIC map produces chaotic sequences in $(-1, 1)$. The value of α is set to 70 in the experiments. Logistic map generates chaotic sequences in (0, 1) and is given by:

$$x_{n+1} = \psi x_n (1 - x_n), 0 < \psi \leq 4, x_n \in (0, 1) \tag{25}$$

4.2.3. Pitch adjustment with decreasing bandwidth and adaptive bandwidth

Due to the significant role of bwin the convergence rate and the search capability suggested by Mahdavi et al. [40] and Zou et al. [71] in creating a balance between exploration and exploitation, a probability P_{bw} is introduced

into GMHS. In the pitch adjustment, the first type bandwidth reduction is employed with the probability of P_{bw}, while with the probability of $1 - P_{bw}$, the second kind of adaptive bandwidth is chosen.

$$bw\,(t)_{1,j} = bw_{1,j,\min} + \left(bw_{1,j,\max} - bw_{1,j,\min}\right) \times \left(\frac{T-t}{T}\right)^{\phi} \tag{26}$$

$$bw_{1,j,\max} = \frac{\mathrm{UB}_j - \mathrm{LB}_j}{2.\mathrm{HMS}} \tag{27}$$

In algorithm 2 (see Appendix D), $bw(t)_{1,j}$ presents the first kind of bandwidth of the jth variable by the t repeat given by Eq. (26). In Eq. (27), $bw_{i,j,\max}$ is the maximum bandwidth value of the jth decision variable given by Eq. (26), ϕ is a positive value, T is the maximal iteration number, and $bw_{i,j,\min}$ is the minimum bandwidth value of the jth decision variable. The $bw(t)_{2,j}$ represents the second kind of bandwidth of the jth decision variable in iteration 1 that changes adaptively, $N(\left|X_{r1,j} - X_{r2,j}\right|)$, $\left|\frac{X_{r1,j}-X_{r2,j}}{10}\right|$ is a value normally distributed with mean $\left|x_{r1,j} - x_{r2,j}\right|$ and standard deviation $\left|x_{r1,j} - x_{r2,j}/10\right|$. The decision variables $x_{r1,j}$ and $x_{r2,j}$ are randomly selected from different harmonies stored in the hm.

4.2.4. Gaussian mutation operator

Although a harmony search algorithm can converge to a global optimum for single-objective optimization, such a convergence may be interrupted in applying the algorithm to solve the MOOPs. This somehow improves the local searchability and global ability. The procedure of the Gaussian mutation operator in the GMHS is described as Algorithm 3 (see Appendix E).

$$\sigma = \frac{\mathrm{UB}_j - \mathrm{LB}_j}{kx} \tag{28}$$

The $N(X_{new,j,\sigma})$ is a value in Algorithm 2, which is normally distributed with a mean $X_{new,j}$ and standard deviation of σ. The σ usually is dependent on the length $\Delta x = \mathrm{UB}_j - \mathrm{LB}_j$ and is computed by Eq. (28). In Eq. (28), kx is a coefficient to control the value of σ and is set equal to 20. According to algorithm 2, the value $x_{new,j}$ of the jth decision variable remains unchanged with the probability $1 - P_{gm}$, while $X_{ew,j}$ is updated by the Gaussian mutation step with the probability p_{gm}. In the Gaussian mutation step, a mutated value $X^{mut}_{new,j}$ is generated by a Gaussian function, and for the value of $X^{mut}_{new,j}$, j is limited in $[\mathrm{LB}_i, \mathrm{UB}_j]$ and then $X_{ew,j}$ is displaced by the mutated value $X^{mut}_{new,j}$.

4.2.5. Updating the hm

The updating of hm in the proposed GMHS algorithm is quite different from in the basic harmony search algorithm. Deb et al. [16] proposed a rapid non-dominant sorting and crowding distance method to be used in updating the hm studied by Sivasubramani and Swarup [60].

4.2.6. Updating the external archive

Deb et al. [16] proposed a crowding distance method which is impractical for problems with more than two objectives. The updating procedure of the external archive is described as Algorithm 4 (see Appendix F) where A_t is the external archive in the iteration t, and hm_{t+1} refers to the best hm for improvisation in the iteration $+1$. N is the maximum size of the external archive. All the Pareto solutions will be retained in the archive A_{t+1}. The procedure of GMHS for MOOPs is described as Algorithm 5 (see Appendix G).

5. Simulation and computational experiments

We evaluate the effectiveness and competitiveness of the EMOHS against four benchmark methods, including NSGAII, multi-objective particle swarm optimization (MOPSO), Pareto envelope-based selection algorithm 2 (PESAII), and multi-objective invasive weed optimization (MOIWO). Deb et al. [16] proposed an upgraded elitist algorithm named NSGA II. In the NSGA II, there are several significant innovations, including a fast non-dominated sorting approach, a quick crowed distance estimation procedure, and a simple crowed comparison operator, which alleviates these difficulties. The MOPSO proposed by Kennedy and Eberhart [30] simulates the flocking behavior of birds. The PESA-II [13] is a classic evolutionary multi-objective optimization with a grid-based fitness assignment

Table 8
The coding of experimental factors and levels.

Factors	Symbol	Coded level		
		−1	0	1
HMCR	X_1	0.97	0.98	0.99
N	X_2	50	100	150
T	X_3	30000	40000	50000
P_C	X_4	0.7	0.8	0.9
HMS	X_5	100	125	150
P_{bw}	X_6	0.6	0.7	0.8
PAR	X_7	0.1	0.15	0.2
P_{gm}	X_8	0.6	0.7	0.8

strategy in environmental selection. The MOIWO, introduced by Kundu et al. [33], simulates the colonizing behavior of the weeds. This behavior is used to search the solution space by increasing the number of iterations and decreasing the distance of the generated weeds from their parent. This competitive elimination process continues until better weeds (solutions) are obtained. We ran the algorithms in the MATLAB 2011 application on a PC with an Intel core i5 processor of 2.30 GHz and 4 GB of RAM memory.

The samples were produced using the following combinations of job number (n) and stage (s), where $n = \{20, 40, 60, 80, 100\}$ and $s = \{2, 4, 8)$. We also generated groups of samples by two parallel machines per stage and a group with several parallel machines at each stage sampled from a discrete uniform distribution in the range [1, 6]. Using a discrete uniform distribution over the interval, we produced the processing times and anticipatory sequence-dependent setup times ([1, 99] & [1.50]). In the case of readiness times, the integers are uniformly distributed between 0 and 100. By uniform distribution in a range of [1, 15], we generated both loading and unloading times. Using the uniform distribution in the range of [1, 30], the traveling times were generated between two subsequent stages. The repeatability chance of every operation was predicted based on an exponential distribution ($\lambda e^{-\lambda}$) with a mean of 0.1. In addition, the rework times of jobs requiring a reworking procedure were generated using a processing time function related to the machine $\left(Round(U(0.3, 0.6) \times p_{i,j})\right)$. The different rates of the factors led to 30 different scenarios.

5.1. Parameter setting

We used the response surface methodology in this article to set the parameters. This methodology, proposed by Box and Wilson [7], is designed to estimate the optimum value of various parameters with an influence on different surfaces of parameters.

Thus, two levels were considered for each effective parameter. At a lower level, the parameters take a value of −1 and +1 at an upper level. Eq. (29) shows the coding of the different levels of the parameters.

$$X_i = \frac{r_i - \left(\frac{h+L}{2}\right)}{\left(\frac{h-L}{2}\right)} \tag{29}$$

Mirhosseini et al. [46] generalized the response surface model to explain the changes in the response variables, which is given below.

$$y = \beta_0 + \sum_{i=1}^{k} \beta_i X_i + \sum_{i=1}^{k} \beta_{ii} X_i^2 + \sum_{i<j} \beta_{ij} X_i X_j + \varepsilon \tag{30}$$

The eight factors $(X_1, X_2, \ldots, X_8)$ were considered as the main variables, including HMCR, N, T, PC, HMS, P_{bw}, PAR, *and* P_{gm}. The values of the high and low parameters are given in Table 8.

We ran each algorithm using different combinations of factors in Table 8 with ten replicates for each. The best make-span was recorded for each implementation. The response variables of the experiment were then computed with the makespan obtained for each instance. The factors with a significant impact on the algorithm are X_1, X_5, X_7, X_2X_3, X_1X_4, X_3X_4, and X_2^2. The mathematical model results are presented in Table 9.

Table 9
Tuned values of the proposed algorithm parameters.

Factor	HMCR	N	T	P_c	HMS	P_{bw}	PAR	P_{gm}
Symbol	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
Best value	0.987	123	40000	0.76	138	0.61	0.182	0.614

5.2. Performance measures

Performance indices were used to compare the results of the multi-objective algorithms quantitatively after normalization of both objective functions. We proposed five indices described by Rabiee et al. [54] and Behnamian et al. [6].

a. Quality metric (QM):
The quality metric is used to simultaneously consider all Pareto solutions obtained by each algorithm, followed by a non-dominant operation. Finally, we defined the quality of each algorithm by a new set of Pareto optimal solutions allocated to every algorithm.

b. Mean ideal distance (MID):
The following formula is used to compute the MID index:

$$\text{MID} = \frac{\sum_{i=1}^{n} c_i}{n} \tag{31}$$

where n is the number of Pareto optimizations and $c_i = \sqrt{f_{1i}^2 + f_{2i}^2}$. It should be noted that the lower the MID, the higher the quality of the algorithm is.

c. The rate of simultaneous outputs of the bi-objective functions (RAS):
Firstly, the ideal point was calculated. Then, the RAS was computed using the following equation:

$$\text{RAS} = \frac{\sum_{i=1}^{n} \left(\frac{|f_{1i} - f_1^{best}|}{f_1^{best}} + \frac{|f_{2i} - f_2^{best}|}{f_2^{best}} \right)}{n} \tag{32}$$

d. Diversification Metric (DM):
This metric can be computed by Eq. (32). In this metric, the algorithm with a higher value has a better capability.

$$\text{DM} = \sqrt{(\max f_{1i} - \min f_{1i})^2 + (\max f_{2i} - \min f_{2i})^2} \tag{33}$$

e. Coverage (C)
The coverage metric proposed by Garcia and Trinh [22], determines the fraction of PF^* captured by the solution PF. This metric has a six-by-six matrix for each value. To find the score for the coverage metric, the average for each row is calculated, and the column for all average scores is normalized as follows:

$$C = \frac{|PF \cap PF^*|}{|PF^*|} = \frac{\left|\{k \in K : \exists i \in I \; such \; that \; PF_{kj}^* = PF_{ij} \; for \; all \; j \in J\}\right|}{|K|} \tag{34}$$

$$PF: = \{F(x) : x \in PS\}, \; PS: = \{x \in X : \exists x' \in X, F(x') \prec F(x)\} \tag{35}$$

5.3. Comparisons among algorithms

We implemented each algorithm according to the job number at different times, and then we compared the results in Table 10.

After calculating the efficiency of the Pareto optimal solutions, we computed the efficiency of each algorithm in terms of DM, MID, RAS, QM, and C.

As shown in Table 11, EMOHS outperforms the other approaches on MID, RAS, QM, and C in terms of mean and number of optimal solutions. However, no significant difference was found among algorithms on DM.

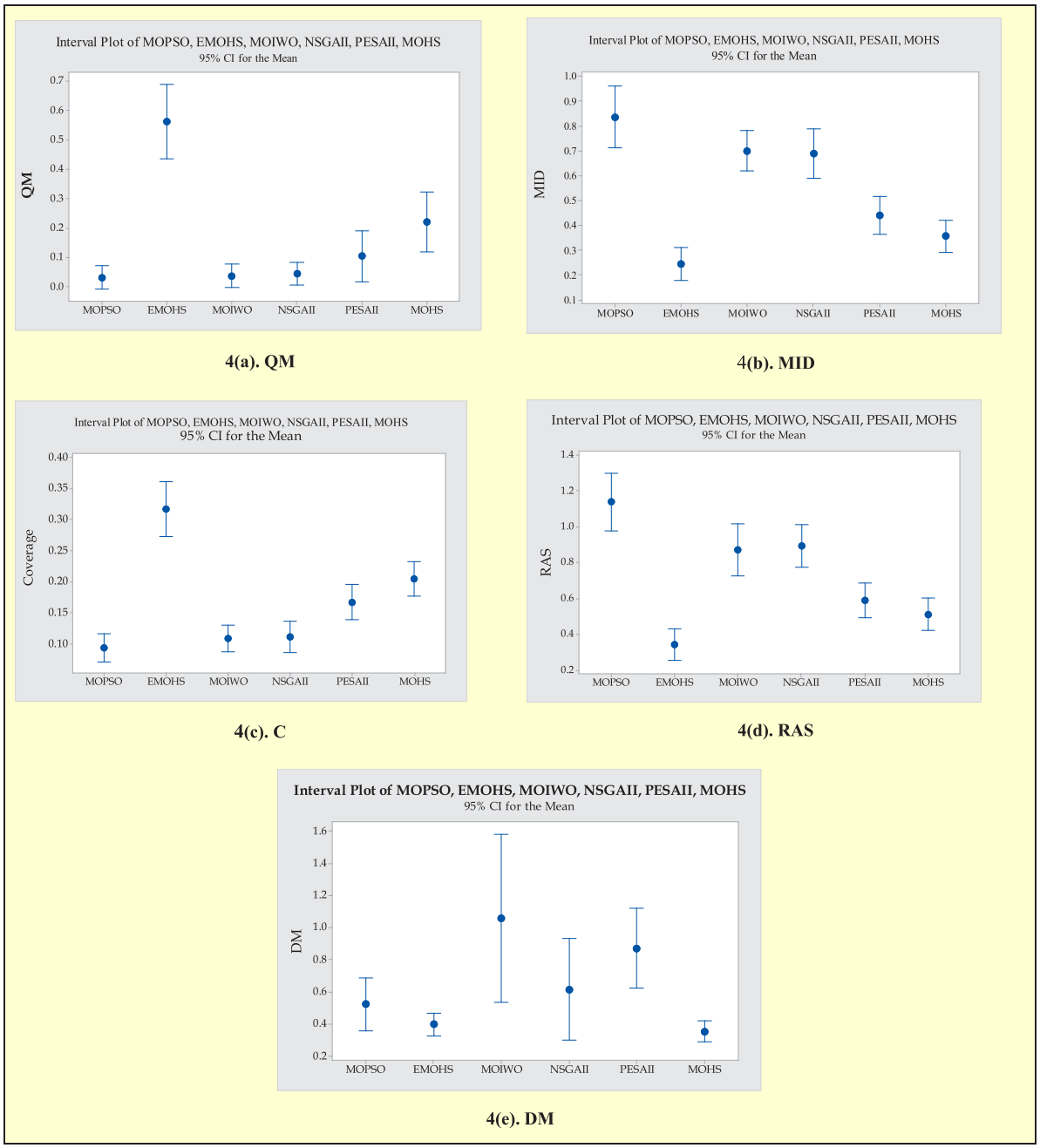


Fig. 4. Means and interval plot for algorithms for each performance measures.

As shown in Fig. 4(a), EMOHS outperforms all other algorithms on the QM (quality) metric since higher values are preferred to lower values. The MOHS algorithm has the next best performance in comparison with the MOPSO, MOIWO, and NSGAII algorithms according to the QM metric. However, there is no statistical significance between MOHS and PESAII on QM. We also could not find any statistical difference among MOIWO, MOPSO, and NSGAII with regards to the QM metric. Fig. 4(b) shows the EMOHS algorithm outperforms all other algorithms on the MID (mean ideal distance) metric. However, this difference is not statistically significant compared with the MOHS algorithm since their CIs overlap. Moreover, the MOHS and PESAII algorithms have the next best performance on

Table 10
Job number and CPU time (second).

Job number	CPU time
20	100
40	500
60	1000
80	2000
100	3000

the MID metric. However, there is no statistical difference between these two algorithms, according to the MID metric.

Fig. 4(c) shows the EMOHS algorithm performs better than the other algorithms on the C (coverage) metric, and this difference is statistically significant. The MOIWO and NSGAII algorithms have similar performance, and there is no statistical difference between them. The MOPSO algorithm performs worse than the other algorithms considering the average performance. Fig. 4(d) indicates the EMOHS algorithm has the best performance among all algorithms. However, its difference with the MOHS algorithm is not statistically significant. The worst performance, according to the RAS metric, belongs to the MOPSO algorithm, the same as the QM and MID metrics. According to the DM (diversification) metric in Fig. 4(e), the MOIWO algorithm performs better than the other algorithms. However, this difference is not statistically significant if we compare it with the MOPSO, NSGAII, and PESAII algorithms. Finally, In contrast with the QM, MID, RAS, and C metrics, the EMOHS and MOHS algorithms have not displayed their superiority according to the DM metric.

6. Conclusion and future research directions

The flexible flow shop scheduling problem is a well-known NP-hard problem composed of flow shops and parallel machines. The transportation times, sequence-dependent setup times, and rework times have been long overlooked in the flexible flow shop scheduling research. Transportation time is often an unavoidable time causing idle times or waiting times in job scheduling. The sequence-dependent setup time plays a crucial role in many industries (e.g., chemical manufacturing) where the cleaning process and intensity are highly dependent on the most recently proceed product and the next product to be processed on that machine. Rework is the transformation of production rejects into re-usable products of the same or lower quality. Rework can be very profitable, especially when disposal costs are high and materials are expensive or limited in availability. The insufficient research on considering these issues and the opportunity to work on a challenging real-world problem motivated us to develop and implement the model proposed in this study. To tackle these concerns, we proposed an enhanced multi-objective algorithm and designed a simulation–optimization framework for implementing the rework process. The proposed algorithm has four attractive and distinctive features, including (i) generating the initial solution using a recently developed constructive heuristics, (ii) employing a Gaussian mutation to enhance the exploration quality, (iii) utilizing a novel K-means clustering approach for improving the quality and diversity of the obtained Pareto solutions, and (iv) applying a response surface methodology for tuning all parameters and increasing reliability.

The EMOHS algorithm is compared with five well-known algorithms, including NSGAII, MOPSO, PESAII, MOHS, and MOIWO. Five metrics of DM, MID, RAS, QM, and C are used for evaluation and comparison purposes. The EMOHS algorithm outperformed all five competing algorithms in four of the five performance metrics except for the DM metric. The MOHS algorithms performed better than the MOPSO, MOIWO, and NSGAII algorithms in all performance metrics. It is worth mentioning that this difference was not significant in some cases. For example, the MOHS and PESAII algorithms do not have significant differences, and the average performance of the MOHS algorithm is slightly better than others. The EMOHS algorithms outperformed other algorithms on the RAS metric, but this difference was not statistically significant. The MOIWO and NSGAII algorithms have similar performance on all performance metrics, and there is no statistical difference among them. They are just different in terms of their average performance. Finally, the MOPSO algorithm has worse performance in comparison with the other algorithms on all metrics except for the DM metric. Possible lines of future research include flexible flow shop scheduling systems with unrelated parallel machines, dedicated machines, and the non-resemble case for the rework procedure and the hybridization of other meta-heuristic algorithms with the EMOHS algorithm. Additional future

Table 11
The results of performance measures (QM, DM, RAS, DM, and C).

	Quality						MID						RAS						DM						Coverage					
	MOPSO	EMOHS	MOIWO	NSGAII	PESAII	MOHS	MOPSO	EMOHS	MOIWO	NSGAII	PESAII	MOHS	MOPSO	EMOHS	MOIWO	NSGAII	PESAII	MOHS	MOPSO	EMOHS	MOIWO	NSGAII	PESAII	MOHS	MOPSO	EMOHS	MOIWO	NSGAII	PESAII	MOHS
20-2-16	0	1	0	0	0	0	1.4142	0.0781	0.7733	0.3587	0.3587	0.214	2	0.0781	0.4193	0.3664	0.3664	0.2214	0	0.47	0.726	2.75	0.715	0.449	0.03	0.5	0.07	0.11	0.11	0.18
20-2-22	0	1	0	0	0	0	1.36	0.2268	0.5665	0.9523	0.4003	0.3846	1.76	0.242	0.7328	1.1705	0.5401	0.4559	0.56	0.377	0.192	3.35	0.82	0.583	0.05	0.36	0.13	0.08	0.19	0.19
20-4-16	0	1	0	0	0	0	1.0167	0.1316	1.2425	0.5921	0.472	0.4476	1.3964	0.1339	1.7375	0.7784	0.6118	0.5905	0	0.341	0.683	0.09	1.66	0.11	0.06	0.51	0.05	0.11	0.13	0.14
20-4-22	0	1	0	0	0	0	0.9939	0.3746	0.495	0.8902	0.495	0.495	1.327	0.4874	0.5772	1.2036	0.5772	0.5772	1.43	0.314	0.354	0.231	0.65	0.51	0.09	0.25	0.18	0.12	0.18	0.18
20-8-16	0	0.1667	0	0	0	0.8333	0.954	0.5299	0.7505	0.7505	0.6245	0.3138	1.2797	0.7297	0.9121	0.9121	0.826	0.3581	0.781	0.343	0.452	0.34	3.3	0.5	0.1	0.2	0.12	0.12	0.15	0.31
20-8-22	0	0.8333	0	0	0	0.1667	0.3685	0.1569	0.3292	0.5724	0.3292	0.2606	0.5027	0.1734	0.4229	0.7489	0.4229	0.2606	0.356	0.243	0.84	2.31	0.605	0.112	0.13	0.32	0.14	0.08	0.14	0.19
40-2-16	0	1	0	0	0	0	1.2625	0	0.9486	0.8314	0.4688	0.3951	1.7757	0	1.2141	1.158	0.6374	0.5043	0.123	0.223	0.44	0.085	0.382	0.38	0.04	0.61	0.05	0.06	0.11	0.13
40-2-22	0	0.3333	0	0.3333	0	0.3333	0.7236	0.208	0.8643	0.4703	0.6812	0.561	1.0334	0.3023	1.1964	0.6711	0.8414	0.7529	0.5618	0.704	0.651	0.07	0.74	0.077	0.1	0.38	0.09	0.18	0.11	0.14
40-4-16	0	0.6667	0	0	0	0.3333	0.6899	0.1485	0.8989	0.6899	0.686	0.1365	1.0703	0.234	1.2237	1.0703	0.9501	0.7174	0.37	0.467	2.97	0.191	1.74	0.12	0.07	0.35	0.06	0.07	0.07	0.38
40-4-22	0	0.5	0	0	0	0.5	0.5353	0.3023	0.8407	1.4142	0.6489	0.432	0.9741	0.6078	1.281	1.364	1.0433	0.826	0	0.634	0.761	0.34	0.34	0.322	0.17	0.32	0.1	0.06	0.14	0.21
40-8-16	0.25	0.5	0	0	0	0.25	0.228	0.361	0.8086	0.5811	0.6111	0.334	0.5908	0.7411	0.9225	0.8026	0.8181	0.6652	0	0.22	0.47	0.14	0.6	0.604	0.3	0.19	0.09	0.11	0.11	0.2
40-8-22	0	0.33	0.166667	0.166667	0.166667	0.166667	0.8026	0.5908	0.456	0.478	0.575	0.518	1.138	0.8374	0.675	0.744	0.783	0.791	0.225	0.315	0.062	0.32	0.92	0.224	0.12	0.15	0.21	0.19	0.16	0.17
60-2-16	0	0.6667	0	0.3333	0	0	1.4142	0.3282	0.5335	0.2676	0.5317	0.4679	1.561	0.3898	0.8327	0.3258	0.6842	0.576	1.148	0.543	3.2	0.49	0.559	0.078	0.05	0.23	0.14	0.28	0.14	0.16
60-2-22	0	0	0	0	0.5	0.5	0.9098	0.4925	0.741	0.648	0.1959	0.2632	1.2261	0.616	0.822	0.718	0.312	0.4963	0.351	0.327	0.32	0.267	0.578	0.47	0.07	0.14	0.09	0.09	0.35	0.26
60-4-16	0.2	0	0.2	0	0	0.6	0.412	0.5845	0.2693	0.8796	0.7875	0.3315	0.386	0.6291	0.3453	1.0924	0.8291	0.363	0.237	0.755	4.56	0.572	0.428	0.42	0.19	0.12	0.29	0.08	0.09	0.23
60-4-22	0	0.5	0.25	0	0	0.25	0.7867	0.5465	0.6601	0.7021	0.8917	0.7758	1.0384	0.6291	0.7586	0.8683	1.193	1.2286	0.492	0.5	4.916	0.503	0.605	0.371	0.15	0.23	0.18	0.16	0.13	0.15
60-8-16	0	1	0	0	0	0	0.79	0	0.56	1.02	0.272	0.095	0.8814	0	0.72	1.221	0.4605	0.2213	0.77	0.497	4.05	0.112	2.447	0.2724	0.04	0.36	0.06	0.03	0.13	0.38
60-8-22	0.5	0.5	0	0	0	0	0.513	0.441	0.935	0.818	0.614	0.75	0.581	0.4605	1.54	1.203	0.6628	0.791	0.84	0.675	2.267	2.07	1.64	0.253	0.2	0.25	0.11	0.13	0.17	0.14
80-2-16	0	0.5	0	0.25	0.25	0	1.074	0.289	0.728	0.213	0.267	0.481	1.102	0.366	0.488	0.1091	0.288	0.452	0.543	0.183	0.471	0.05	0.468	0.293	0.05	0.23	0.08	0.28	0.24	0.12
80-2-22	0	0.5	0.5	0	0	0	1.15	0.103	0.307	0.6872	0.56	0.415	1.182	0.2102	0.332	0.7293	0.653	0.493	0	0.553	0.23	0.628	0.72	0.19	0.04	0.48	0.21	0.07	0.08	0.12
80-4-16	0	0.3334	0	0	0.5	0.1667	0.501	0.2047	0.8571	1.0441	0.389	0.246	0.648	0.245	0.7071	1.353	0.5673	0.381	0.972	0.579	0.226	0.428	0.52	0.229	0.12	0.33	0.08	0.06	0.16	0.25
80-4-22	0	0.6	0	0	0.1	0.3	1.2987	0.154	0.743	1.032	0.376	0.452	1.1559	0.193	0.532	0.9336	0.2378	0.276	1.678	0.167	0.249	0.274	0.321	0.625	0.05	0.45	0.09	0.06	0.2	0.15
80-8-16	0	0.25	0	0.25	0.5	0	0.7852	0.251	0.66	0.167	0.199	0.365	1.657	0.541	1.425	0.308	0.318	0.624	0.57	0.416	0.621	0.964	0.86	0.424	0.06	0.2	0.11	0.28	0.23	0.12
80-8-22	0	0	0	0	0	1	0.538	0.127	0.457	0.6345	0.325	0	0.618	0.3214	0.735	1.167	0.435	0	1.34	0.135	0.212	0.21	0.65	0.565	0.07	0.35	0.1	0.06	0.13	0.29
100-2-16	0	0	0	0	1	0	0.459	0.212	0.993	0.716	0	0.383	0.5729	0.3214	1.2043	0.834	0	0.435	0.455	0.121	0.343	0.35	0.25	0.2	0.11	0.24	0.05	0.07	0.4	0.13
100-2-22	0	0.6	0	0	0	0.4	1.0932	0.135	0.827	0.675	0.149	0.29	1.4673	0.273	1.228	0.997	0.441	0.761	0.437	0.337	0.269	0.06	0.817	0.228	0.04	0.35	0.06	0.07	0.31	0.17
100-4-16	0	1	0	0	0	0	0.863	0	0.658	0.796	0.392	0.256	1.461	0	0.983	1.272	0.6217	0.457	0.589	0.6404	0.249	0.267	0.84	0.501	0.06	0.43	0.09	0.06	0.14	0.22
100-4-22	0	0.7	0	0	0.1	0.2	0.926	0.161	0.578	0.742	0.343	0.272	1.691	0.2378	0.892	1.089	0.662	0.458	0	0.216	0.379	0.3671	0.355	0.31	0.06	0.37	0.1	0.08	0.17	0.22
100-8-16	0	1	0	0	0	0	0.365	0	0.769	0.587	0.272	0.187	0.797	0	1.143	0.922	0.461	0.359	0.367	0.069	0.49	0.552	0.855	0.691	0.12	0.31	0.06	0.08	0.17	0.26
100-8-22	0	0.4	0	0	0	0.6	0.894	0.174	0.746	0.448	0.286	0.141	1.243	0.301	0.1.165	0.654	0.389	0.215	0.481	0.55	0.08	0.064	0.77	0.489	0.05	0.29	0.06	0.1	0.16	0.34

Note; $v =$ machine distribution function is variable (between 1 and 6) and $c =$ machine distribution function is constant (i.e. equal to 2).

research opportunities include adding energy consumption to the current setting and see its effect on the optimal solution and its interaction with other features of the model.

To the best of our knowledge, this research is the first study to consider loading, transportation, and unloading times in flexible flow shop scheduling. However, we assumed adequate labor for loading and unloading products, which can be relaxed in future research by considering a total available labor constraint for loading and unloading jobs. In addition, developing a lower bound for the delivery date-related objective functions could be considered as another stream of future research. Furthermore, testing different failure distribution types and rework times, along with their impact on the production schedule, is another interesting topic for future research. Finally, developing new algorithms with machine learning can enhance the search process and improve the solution quality in multi-objective algorithms.

Acknowledgment

The authors would like to thank the anonymous reviewers and the editor for their insightful comments and suggestions. Dr. Madjid Tavana is grateful for the partial support he received from the Czech Science Foundation (GAČR 19-13946S) for this research.

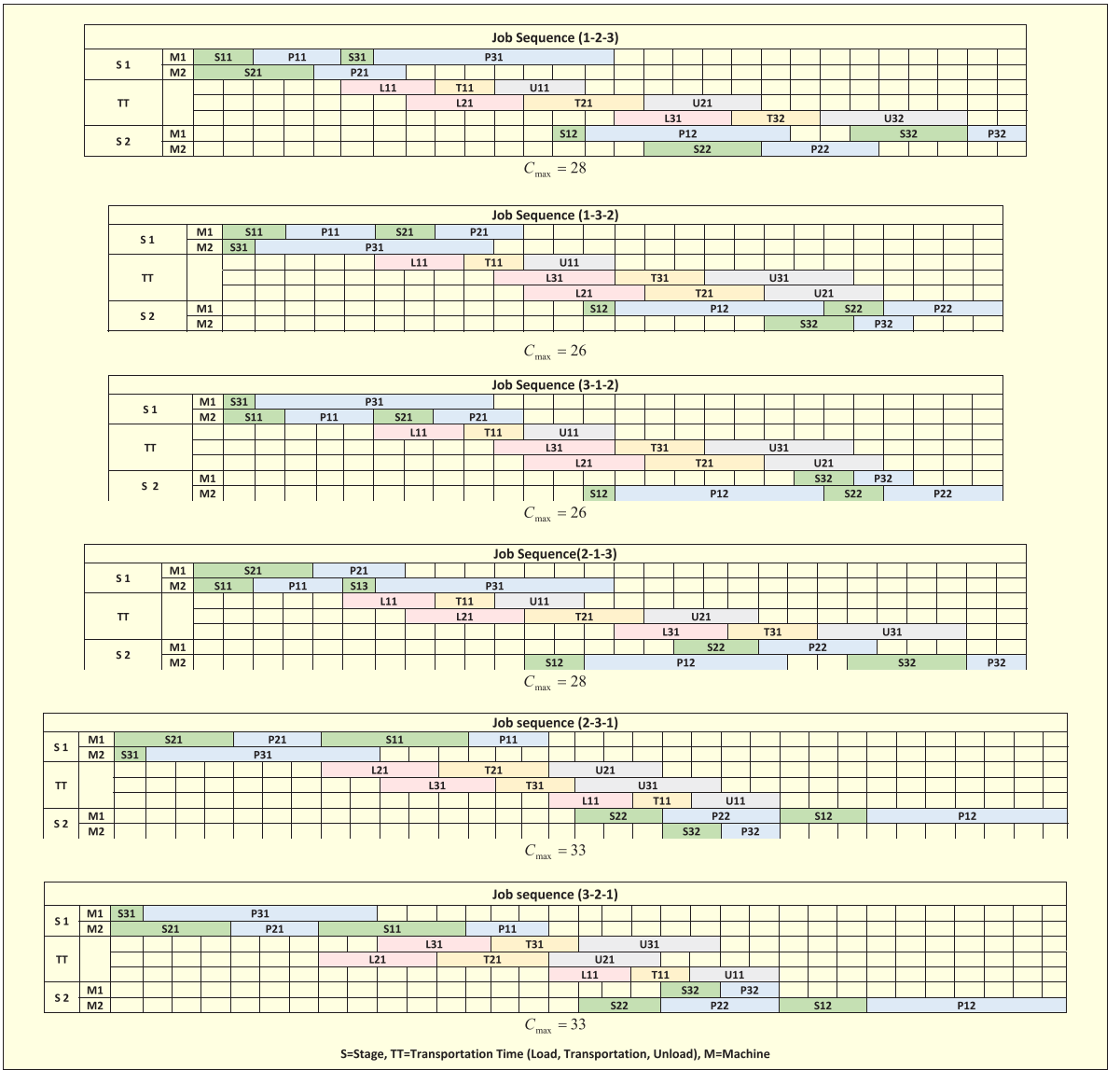
Appendix A

A comprehensive list of all acronyms and abbreviations

bw	*Distance bandwidth*
DM	*Diversification Metric*
GMHS	*Harmony search algorithm with Gaussian mutation*
HFS	*Hybrid flow shop*
hm	*Harmony memory*
HMCR	*Harmony memory considering a rate*
ICMIC	*Iterative chaotic map with infinite collapses*
MID	*Mean ideal distance*
MOHS	*Multi-objective harmony search*
MOIWO	*Multi-objective invasive weed optimization*
MOOP	*Multi-objective optimization problems*
MOPSO	*Multi-objective particle swarm optimization*
NP	*Nondeterministic polynomial time*
NSGAII	*Non-dominated Sorting Genetic Algorithm-II*
PAR	*Pitch adjusting rate*
PESAII	*Pareto envelope-based selection algorithm 2*
QM	*Quality metric*
RAS	*Rate of simultaneous outputs of the bi-objective functions*
SDST	*Sequence-dependent setup times*
SNS	*The spread of Pareto optimization*

Appendix B

See below.



Appendix C

See Algorithm 1.

Algorithm 1: The pseudo-code of Procedure MCH-12

$\Gamma := \{\gamma_1,...,\gamma_i,...,\gamma_n\}$: *Jobs orded by non – increasing sum of processing times*;
$\Pi^{(1)} := \{\gamma_1\}$;
for $k = 2$ *to* n *do*
  $S^k := k.x$;
  Insert job γ_k *in any possible position* j *of* $\Pi^{(k-1)}$, *with* $j \in \{1,....,k\}$;
  $\Pi^{(k)}$:= *sequence obtained by inserting* γ_k *in the position of* $\Pi^{(k-1)}$ *with lowest* OF'. *Let* F^b *denote such value of the objective function*;
  $r_s^{(k)}$ (*with* $s \in \{1,....,S^{(k)}\}$) := *job before* γ_k, *after testing* γ_k *in the sth position with lowest* OF';
  if $k \succ 2$ *then*
    for $s = 1$ *to* $S^{(k-1)}$ *do*
      $\Phi^{(k)}$:= *permutation obtained by removing job* γ_{k-1} *from* $\Pi^{(k)}$ *and re – insert it after job* $r_s^{(k-1)}$. *Let* F^m *be the value of* OF';
      if $F^m \prec F^b$ *then*
        $F^b = F^m$;
        $\Pi^{(k)} = \Phi^{(k)}$;
      end
    end
  end
end
  for $k = 1$ *to* n *do*
    Ω:= *solution obtained by considering* $\Pi^{(n)}$ *as the sequence in the first stage, and by generating a random sequence for the other stages as follows: Once a machine becomes free, the next job to be processed is randomly choosen among the jobs in the queue. Let* F^s *denote the objective function obtained*;
    if $F^s \prec F^b$ *then*
      $F^b = F^s$;
    end
  end
end

Appendix D

See Algorithm 2.

Algorithm 2: The pseudo-code of pitch adjustment in the GMHS

Commute PAR according to Logistic map
if $rand() \le PAR$ *then*
  if $rand() \le P_{bw}$ *then*
    Generate chaotic vaiable γ_1 *according to ICMIC map*
    $x_{new,j} = x_{new,j} + bw(t)_{1,j} \times \gamma_1$ *// the first type of pitch adjustment*
else
    Generate chaotic variable γ_2 *according to ICMIC map*
    $r1$=*randomly generated positive integers in the region of* $[1, HMS]$
   while $r1 = r2$ *do*
    $r2 =$ *randomly generated positive integers in the region of* $[1, HMS]$
   end while
    $bw(t)_{2,j} = N(|x_{r1,j} - x_{r2,j}|, |x_{r1,j} - x_{r2,j}|/10)$
    $x_{new,j} = x_{new,j} + bw(t)_{2,j} \times \gamma_2$ *// the second type of pitch adjustment*
end if
if $x_{new,j} > UB_j$ *then*
    $x_{new} = UB_j$
end if
if $x_{new,j} < LB_j$ *then*
    $x_{new,j} = LB_j$
end if
end if

Appendix E

See Algorithm 3.

Algorithm 3: The pseudo-code of Gaussian mutation

```
for j = l : n do
  if rand() < P_gm then
    x^mut_new,j = N(x_new,j, σ)
    if x^mut_new,j < LB_j then
      x^mut_new,j = LB_j
    end if
    if x^mut_new,j > UB_j then
      x^mut_new,j > UB_j
    end if
    x_new,j = x^mut_new,k
    else
      x_new,j = x_new,j
    end if
  end for
```

Appendix F

See Algorithm 4.

Algorithm 4: The pseudo-code of updating the external archive in the GMHS

```
A_{t+1} = all non-dominated solutions found in A_t ⋃ hm_{t+1}
M = the size of A_{t+1}
while M > N do
  if the problem is of two objectives then all solutions in A_{t+1} are assigned a
  crowding distance value using crowding distance metric (Deb et al., 2002).
  The solution with the smallest crowding distance value is deleted.
else
  An archive truncation procedure in Zitzler et al. (2001) is adopted, the solution
  which has the minimum distance to another solution is deleted.
 end if
end while
```

Appendix G

See Algorithm 5.

Algorithm 5: The pseudo-code GMHS for MOPS

Input : $HMS, N(archive\ size), T(maximal\ iteration\ number)$.
$HMCR, P, P_{bw}, bw_{1,j,\min}$
output : $A(non-dominated\ set)$
Step 1 : *Intialize the algorithm parameters, set the iteration number* $t = 0$ *and generate an initial harmony memory* hm_t *based on the problem range and constraints if any, evalute all harmonies in* hm_t *and create the empty external archive* $A_t = \varphi$.
Step 2 : *Improvise a new hm. Perform improvisation and Gaussian mutation operator in the GMHS for HMS times to generate a new harmony memory* hm_t^{new} *and evaluate objectives functions of each harmony in* hm_t^{new}.
Step 3 : *Update hm. Select the best harmony memory* hm_{t+1} *from the combined harmony memory* $hm_t \bigcup hm_t^{new}$ *as given in Section* 4.5 *for the next iteration.*
Step 4 : *Update external archive : Get an updated external archive* A_{t+1} *by executing Algorithm* 4 *in Section* 4.4.5.
Step 5 : *Check termination criterion. If* $t \geq T$ *is reached, set A to the set of solutions expressed by the non-dominated solutions in* A_{t+1}, *output the non − dominated set A and terminate the algorithm. Otherwise, increase iteration counter* $(t = t + 1)$ *and go to Step* 2.
return A

References

[1] L. Adler, N. Fraiman, E. Kobacker, M. Pinedo, J.C. Plotnicoff, T.P. Wu, BPSS: a scheduling support system for the packaging industry, Oper. Res. 41 (4) (1993) 641–648.
[2] F.S. de Almida, Optimization of laminated composite structures using harmony search Dalgorithm, Compos. Struct. 221 (2019) 110852.
[3] H. Bargaoui, O.B. Driss, Multi-agent model based on tabu search for the permutation flow shop scheduling problem, in: Distributed Computing and Artificial Intelligence, 11th International Conference, Springer, Cham, 2014, pp. 519–527.
[4] J. Behnamian, S.M.T. Fatemi Ghomi, M. Zandieh, Development of a hybrid metaheuristic to minimise earliness and tardiness in a hybrid flowshop with sequence-dependent setup times, Int. J. Prod. Res. 48 (5) (2010) 1415–1438.
[5] J. Behnamian, S.F. Ghomi, Hybrid flowshop scheduling with machine and resource-dependent processing times, Appl. Math. Model. 35 (3) (2011) 1107–1123.
[6] J. Behnamian, S.F. Ghomi, M. Zandieh, A multi-phase covering pareto-optimal front method to multi-objective scheduling in a realistic hybrid flowshop using a hybrid metaheuristic, Expert Syst. Appl. 36 (8) (2009) 11057–11069.
[7] G.E. Box, K.B. Wilson, On the experimental attainment of optimum conditions, J. R. Stat. Soc. Ser. B Stat. Methodol. 13 (1) (1951) 1–45.
[8] S.C. Chang, D.Y. Liao, Scheduling flexible flow shops with no setup effects, IEEE Trans. Robot. Autom. 10 (2) (1994) 112–122.
[9] C.L. Chen, Iterated population-based VND algorithms for single-machine scheduling with sequence-dependent setup times, Soft Comput. (2018) 1–15.
[10] J. Chen, Q.K. Pan, J.Q. Li, Harmony search algorithm with dynamic control parameters, Appl. Math. Comput. 219 (2) (2012) 592–604.
[11] J. Chen, Q.K. Pan, L. Wang, J.Q. Li, A hybrid dynamic harmony search algorithm for identical parallel machines scheduling, Eng. Optim. 44 (2) (2012) 209–224.
[12] C.Y. Cheng, K.C. Ying, S.F. Li, Y.C. Hsieh, Minimizing makespan in mixed no-wait flowshops with sequence-dependent setup times, Comput. Ind. Eng. 130 (2019) 338–347.
[13] D.W. Corne, N.R. Jerram, J.D. Knowles, M.J. Oates, PESA-II: Region-based selection in evolutionary multiobjective optimization, in: Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation, 2001, pp. 283–290.
[14] N.C.O. Da Silva, C.T. Scarpin, J.E. Pécora Jr., A. Ruiz, Online single machine scheduling with setup times depending on the jobs sequence, Comput. Ind. Eng. 129 (2019) 251–258.
[15] X. Dai, X. Yuan, L. Wu, A novel harmony search algorithm with Gaussian mutation for multi-objective optimization, Soft Comput. (2015) 1–19.
[16] K. Deb, A. Pratap, S. Agarwal, T.A.M.T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.
[17] M. Dios, V. Fernandez-Viagas, J.M. Framinan, Efficient heuristics for the hybrid flow shop scheduling problem with missing operations, Comput. Ind. Eng. 115 (2018) 88–99.

[18] F. Dugardin, F. Yalaoui, L. Amodeo, New multi-objective method to solve reentrant hybrid flow shop scheduling problem, European J. Oper. Res. 203 (1) (2010) 22–31.
[19] M. Ebrahimi, S.F. Ghomi, B. Karimi, Hybrid flow shop scheduling with sequence dependent family setup time and uncertain due dates, Appl. Math. Model. 38 (9) (2014) 2490–2504.
[20] V. Fernandez-Viagas, J.M. Molina-Pariente, J.M. Framinan, New efficient constructive heuristics for the hybrid flowshop to minimise makespan: A computational evaluation of heuristics, Expert Syst. Appl. 114 (2018) 345–356.
[21] V. Fernandez-Viagas, P. Perez-Gonzalez, J.M. Framinan, Efficiency of the solution representations for the hybrid flow shop scheduling problem with makespan objective, Comput. Oper. Res. 109 (2019) 77–88.
[22] S. Garcia, C.T. Trinh, Comparison of multi-objective evolutionary algorithms to solve the modular cell design problem for novel biocatalysis, Processes 7 (6) (2019) 361.
[23] Z.W. Geem, J.H. Kim, G.V. Loganathan, A new heuristic optimization algorithm: harmony search, Simulation 76 (2) (2001) 60–68.
[24] S.M. Gholami-Zanjani, M. Hakimifar, N. Nazemi, F. Jolai, Robust and fuzzy optimisation models for a flow shop scheduling problem with sequence dependent setup times: A real case study on a PCB assembly company, Int. J. Comput. Integr. Manuf. 30 (6) (2017) 552–563.
[25] A.G.P. Guinet, M.M. Solomon, Scheduling hybrid flow shops to minimize maximum tardiness or maximum completion time, Int. J. Prod. Res. 34 (6) (1996) 1643–1654.
[26] D. He, C. He, L.G. Jiang, H.W. Zhu, G.R. Hu, Chaotic characteristics of a one-dimensional iterative map with infinite collapses, IEEE Trans. Circuits Syst. I 48 (7) (2001) 900–906.
[27] F.J. Hwang, B.M. Lin, Two-stage flexible flow shop scheduling subject to fixed job sequences, J. Oper. Res. Soc. 67 (3) (2016) 506–515.
[28] F. Jabbarizadeh, M. Zandieh, D. Talebi, Hybrid flexible flowshops with sequence-dependent setup times and machine availability constraints, Comput. Ind. Eng. 57 (3) (2009) 949–957.
[29] N. Karimi, M. Zandieh, H.R. Karamooz, Bi-objective group scheduling in hybrid flexible flowshop: a multi-phase approach, Expert Syst. Appl. 37 (6) (2010) 4024–4032.
[30] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95-International Conference on Neural Networks, Vol. 4, IEEE, 1995, pp. 1942–1948.
[31] A. Khare, S. Agrawal, Scheduling hybrid flowshop with sequence-dependent setup times and due windows to minimize total weighted earliness and tardiness, Comput. Ind. Eng. 135 (2019) 780–792.
[32] S. Kulluk, L. Ozbakir, A. Baykasoglu, Training neural networks with harmony search algorithms for classification problems, Eng. Appl. Artif. Intell. 25 (1) (2012) 11–19.
[33] D. Kundu, K. Suresh, S. Ghosh, S. Das, B.K. Panigrahi, S. Das, Multi-objective optimization with artificial weed colonies, Inform. Sci. 181 (12) (2011) 2441–2454.
[34] M.E. Kurz, R.G. Askin, Scheduling flexible flow lines with sequence-dependent setup times, European J. Oper. Res. 159 (1) (2004) 66–82.
[35] K.S. Lee, Z.W. Geem, A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice, Comput. Methods Appl. Mech. Engrg. 194 (36) (2005) 3902–3933.
[36] D. Li, X. Meng, Q. Liang, J. Zhao, A heuristic-search genetic algorithm for multi-stage hybrid flow shop scheduling with single processing machines and batch processing machines, J. Intell. Manuf. 26 (5) (2015) 873–890.
[37] S.W. Lin, J.N. Gupta, K.C. Ying, Z.J. Lee, Using simulated annealing to schedule a flowshop manufacturing cell with sequence-dependent family setup times, Int. J. Prod. Res. 47 (12) (2009) 3205–3217.
[38] C. Lu, L. Gao, X. Li, Q. Pan, Q. Wang, Energy-efficient permutation flow shop scheduling problem using a hybrid multi-objective backtracking search algorithm, J. Cleaner Prod. 144 (2017) 228–238.
[39] Q. Luo, X. Yang, Y. Zhou, Nature-inspired approach: An enhanced moth swarm algorithm for global optimization, Math. Comput. Simulation 159 (2019) 57–92.
[40] M. Mahdavi, M. Fesanghary, E. Damangir, An improved harmony search algorithm for solving optimization problems, Appl. Math. Comput. 188 (2) (2007) 1567–1579.
[41] M.K. Marichelvam, M. Geetha, Application of novel harmony search algorithm for solving hybrid flow shop scheduling problems to minimise makespan, Int. J. Ind. Syst. Eng. 23 (4) (2016) 467–481.
[42] M.K. Marichelvam, Ö. Tosun, M. Geetha, Hybrid monkey search algorithm for flow shop scheduling problem under makespan and total flow time, Appl. Soft Comput. 55 (2017) 82–92.
[43] M.M. Mazdeh, M. Rostami, A branch-and-bound algorithm for two-machine flow-shop scheduling problems with batch delivery costs, Int. J. Syst. Sci. Oper. Logist. 1 (2) (2014) 94–104.
[44] R.K. Meena, M. Jain, S.S. Sanga, A. Assad, Fuzzy modeling and harmony search optimization for machining system with general repair, standby support and vacation, Appl. Math. Comput. 361 (2019) 858–873.
[45] T. Meng, Q.K. Pan, J.Q. Li, H.Y. Sang, An improved migrating birds optimization for an integrated lot-streaming flow shop scheduling problem, Swarm Evol. Comput. 38 (2018) 64–78.
[46] H. Mirhosseini, C.P. Tan, N.S. Hamid, S. Yusof, B.H. Chern, Characterization of the influence of main emulsion components on the physicochemical properties of orange beverage emulsion using response surface methodology, Food Hydrocolloids 23 (2) (2009) 271–280.
[47] B. Naderi, R. Ruiz, M. Zandieh, Algorithms for a realistic variant of flowshop scheduling, Comput. Oper. Res. 37 (2) (2010) 236–246.
[48] B. Naderi, M. Zandieh, V. Roshanaei, Scheduling hybrid flowshops with sequence dependent setup times to minimize makespan and maximum tardiness, Int. J. Adv. Manuf. Technol. 41 (11–12) (2009) 1186–1198.

[49] H.E. Nouri, O.B. Driss, K. Ghédira, A classification schema for the job shop scheduling problem with transportation resources: state-of-the-art review, in: Artificial Intelligence Perspectives in Intelligent Systems, Springer, Cham, 2016, pp. 1–11.
[50] H.E. Nouri, O.B. Driss, K. Ghédira, Hybrid metaheuristics for scheduling of machines and transport robots in job shop environment, Appl. Intell. 45 (3) (2016) 808–828.
[51] H.E. Nouri, O.B. Driss, K. Ghédira, Simultaneous scheduling of machines and transport robots in flexible job shop environment using hybrid metaheuristics based on clustered holonic multiagent model, Comput. Ind. Eng. 102 (2016) 488–501.
[52] H.E. Nouri, O.B. Driss, K. Ghédira, Controlling a single transport robot in a flexible job shop environment by hybrid metaheuristics, in: Transactions on Computational Collective Intelligence XXVIII, Springer, Cham, 2018, pp. 93–115.
[53] M. Pinedo, Scheduling: Theory, Algorithms, and Systems, 2002.
[54] M. Rabiee, M. Zandieh, P. Ramezani, Bi-objective partial flexible job shop scheduling problem: NSGA-II, NRGA, MOGA and PAES approaches, Int. J. Prod. Res. 50 (24) (2012) 7327–7342.
[55] M. Rani, H. Garg, S.P. Sharma, Cost minimization of butter-oil processing plant using artificial bee colony technique, Math. Comput. Simulation 97 (2014) 94–107.
[56] I. Ribas, R. Leisten, J.M. Framiñan, Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective, Comput. Oper. Res. 37 (8) (2010) 1439–1454.
[57] R. Ruiz, J.A. Vázquez-Rodríguez, The hybrid flow shop scheduling problem, European J. Oper. Res. 205 (1) (2010) 1–18.
[58] I. Saad, S. Hammadi, M. Benrejeb, P. Borne, Choquet integral for criteria aggregation in the flexible job-shop scheduling problems, Math. Comput. Simulation 76 (5–6) (2008) 447–462.
[59] A. Sioud, C. Gagné, Enhanced migrating birds optimization algorithm for the permutation flow shop problem with sequence dependent setup times, European J. Oper. Res. 264 (1) (2018) 66–73.
[60] S. Sivasubramani, K.S. Swarup, Multi-objective harmony search algorithm for optimal power flow problem, Int. J. Electr. Power Energy Syst. 33 (3) (2011) 745–752.
[61] B. Sokolov, D. Ivanov, S.A. Potryasaev, Flexible flow shop scheduling for continuous production, Int. J. Serv. Comput. Oriented Manuf. 2 (2) (2016) 189–203.
[62] T.H. Tran, K.M. Ng, A hybrid water flow algorithm for multi-objective flexible flow shop scheduling problems, Eng. Optim. 45 (4) (2013) 483–502.
[63] H. Wang, Flexible flow shop scheduling: optimum, heuristics and artificial intelligence solutions, Expert Syst. 22 (2) (2005) 78–85.
[64] B. Yagmahan, M.M. Yenisey, Ant colony optimization for multi-objective flow shop scheduling problem, Comput. Ind. Eng. 54 (3) (2008) 411–420.
[65] B. Yagmahan, M.M. Yenisey, A multi-objective ant colony system algorithm for flow shop scheduling problem, Expert Syst. Appl. 37 (2) (2010) 1361–1368.
[66] A.J. Yu, J. Seif, Minimizing tardiness and maintenance costs in flow shop scheduling by a lower-bound-based GA, Comput. Ind. Eng. 97 (2016) 26–40.
[67] X. Yuan, Y. Yuan, Y. Zhang, A hybrid chaotic genetic algorithm for short-term hydro system scheduling, Math. Comput. Simulation 59 (4) (2002) 319–327.
[68] M. Zandieh, N. Karimi, An adaptive multi-population genetic algorithm to solve the multi-objective group scheduling problem in hybrid flexible flowshop with sequence-dependent setup times, 22 (6) (2011) 979–989.
[69] B. Zhang, Q.K. Pan, L. Gao, X.Y. Li, L.L. Meng, K.K. Peng, A multiobjective evolutionary algorithm based on decomposition for hybrid flowshop green scheduling problem, Comput. Ind. Eng. (2019).
[70] Q. Zhu, X. Tang, Y. Li, M.O. Yeboah, An improved differential-based harmony search algorithm with linear dynamic domain, Knowl.-Based Syst. (2019).
[71] D. Zou, L. Gao, J. Wu, S. Li, Novel global harmony search algorithm for unconstrained problems, Neuro Comput. 73 (16) (2010) 3308–3318.