Machine Learning with Applications 11 (2023) 100445

Contents lists available at ScienceDirect

Machine Learning with Applications

journal homepage: www.elsevier.com/locate/mlwa

A bi-objective hybrid vibration damping optimization model for synchronous flow shop scheduling problems

Madjid Tavana [a,b,*], Vahid Hajipour [c,d], Mohammad Alaghebandha [e], Debora Di Caprio [f]

[a] Business Systems and Analytics Department, Distinguished Chair of Business Analytics, La Salle University, Philadelphia, PA 19141, United States
[b] Business Information Systems Department, Faculty of Business Administration and Economics, University of Paderborn, D-33098 Paderborn, Germany
[c] Department of Industrial Engineering, Faculty of Engineering and Architecture, Altinbas University, 34128, Mahmutbey, Istanbul, Turkey
[d] Department of Industrial Engineering, Faculty of Engineering, Islamic Azad University, West Tehran Branch, Tehran, Iran
[e] Department of Industrial Engineering, Faculty of Engineering, Kharazmi University, Tehran, Iran
[f] Department of Economics and Management, University of Trento, Trento, Italy

ARTICLE INFO

Keywords:
Multi-objective optimization
Synchronous flow shop scheduling
Meta-heuristics
Epsilon-constraints

ABSTRACT

Flow shop scheduling deals with the determination of the optimal sequence of jobs processing on machines in a fixed order with the main objective consisting of minimizing the completion time of all jobs (makespan). This type of scheduling problem appears in many industrial and production planning applications. This study proposes a new bi-objective mixed-integer programming model for solving the synchronous flow shop scheduling problems with completion time. The objective functions are the total makespan and the sum of tardiness and earliness cost of blocks. At the same time, jobs are moved among machines through a synchronous transportation system with synchronized processing cycles. In each cycle, the existing jobs begin simultaneously, each on one of the machines, and after completion, wait until the last job is completed. Subsequently, all the jobs are moved concurrently to the next machine. Four algorithms, including non-dominated sorting genetic algorithm (NSGA II), multi-objective simulated annealing (MOSA), multi-objective particle swarm optimization (MOPSO), and multi-objective hybrid vibration-damping optimization (MOHVDO), are used to find a near-optimal solution for this NP-hard problem. In particular, the proposed hybrid VDO algorithm is based on the imperialist competitive algorithm (ICA) and the integration of a neighborhood creation technique. MOHVDO and MOSA show the best performance among the other algorithms regarding objective functions and CPU Time, respectively. Thus, the results from running small-scale and medium-scale problems in MOHVDO and MOSA are compared with the solutions obtained from the epsilon-constraint method. In particular, the error percentage of MOHVDO's objective functions is less than 2% compared to the epsilon-constraint method for all solved problems. Besides the specific results obtained in terms of performance and, hence, practical applicability, the proposed approach fills a considerable gap in the literature. Indeed, even though variants of the aforementioned meta-heuristic algorithms have been largely introduced in multi-objective environments, a simultaneous implementation of these algorithms as well as a compared study of their performance when solving flow shop scheduling problems has been so far overlooked.

1. Introduction

Scheduling problems involve optimizing specified objectives and allocating limited resources to jobs within an explicit time. The resources generally include the workforce, machines, raw materials, and service points, among others. The jobs can comprise serving clients, manufacturing processes, or delivering goods. The objectives could be maximizing the number of orders that satisfy the delivery date, minimizing the completion time of jobs (Cmax or makespan), minimizing the maximum tardiness of jobs, or minimizing the mean service time.

In a flow shop manufacturing environment, there are m machines, and all jobs are processed according to a common processing flow. Each machine can only process one job at a time, and an unlimited capacity for intermediate buffers is usually assumed; that is, a job can be kept

* Corresponding author at: Business Systems and Analytics Department, Distinguished Chair of Business Analytics, La Salle University, Philadelphia, PA 19141, United States.
E-mail addresses: tavana@lasalle.edu (M. Tavana), vahid.hajipour@altinbas.edu.tr (V. Hajipour), m.alaghebandha@gmail.com (M. Alaghebandha), debora.dicaprio@unitn.it (D. Di Caprio).
URLs: http://tavana.us/ (M. Tavana), http://vhajipour.com/ (V. Hajipour).

https://doi.org/10.1016/j.mlwa.2022.100445
Received 26 September 2022; Received in revised form 26 November 2022; Accepted 21 December 2022
Available online 26 December 2022

for a limitless amount of time. The problem is deciding each machine's processing order (Huang, 2008).

More precisely, the permutation flow shop problem with n jobs and m machines is generally described as follows. Each one of the n given jobs is to be consecutively processed on machines 1 to m, whereas the processing time ($p_{j,i}$) of job j on machine i is predetermined. The sequence in which the jobs are to be processed is the same for all the machines, while the objective function is to obtain a permutation of jobs that minimizes the makespan, $C_{\max} = \max_{j=1,\ldots,n} C_j$, where C_j is the finishing time of processing job j (the finishing time of the last operation of job j on machine m). (Huang, 2008).

The material handling system can be conveyors, robots, rotary tables, cranes, automated guided vehicles, and carriers (Huang, 2008). The rotary table is encompassed through the processing machines and loading/unloading station, whereas jobs should be loaded up on the table before being processed (Milacron, 1989; Waldherr & Knust, 2014, 2016).

1.1. Aim and challenges

This study aims at designing a suitable formulation and effective solution method for the flow shop scheduling problem with synchronous material transfer considering makespan and the sum of tardiness and earliness cost as two objectives.

As mentioned above, the necessity of formulating and solving this kind of problems can easily arise in robotic cells or machining centers. In such an environment, a job can block a machine while waiting for the robot to pick it up and transfer it to the following step (Ribas & Companys, 2015). In particular, one can consider the case where, in each cycle, the start time of all the jobs on the corresponding machines is synchronous, and after being processed, all of them must stay until when the final job is completed. Subsequently, all jobs are shifted to the succeeding machine simultaneously (Kampmeyer et al., 2016; Waldherr & Knust, 2016).

Thus, the flow shop scheduling problem can be interpreted as a permutation flow shop without preemption, which transmits a job from one machine to another at a time. The following machine's job processing starts when processing the whole job on the current machine is over. If the required time of processing a job on a machine is less than the maximum processing time, this machine will be unoccupied up to the next transfer.

We are also assuming blocking. Thus, there are no buffers among the machines and intermediate queues of jobs waiting in the production system for their following operations are not permitted (Grabowski & Pempera, 2007).

For a better and intuitive understanding of the proposed problem, an example of flow shop with synchronous movement is provided below.

Example: Consider a flow shop with three machines and five jobs where the processing time ($p_{j,i}$) of job j on machine i is predetermined and assigned according to the following matrix:

$$\begin{array}{l} \\ \text{processing time} \\ p_{j,i} \\ \text{on machine } i \end{array} \begin{array}{c} \text{job} \\ j \\ \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{Bmatrix} p_{j,1} \\ p_{j,2} \\ p_{j,3} \end{Bmatrix} \begin{bmatrix} 3 & 1 & 3 & 5 & 2 \\ 1 & 3 & 2 & 1 & 4 \\ 1 & 1 & 5 & 5 & 4 \end{bmatrix} \end{array}$$

Consider the job sequence (3, 4, 2, 1, 5). A scheduled for this sequence is represented in Fig. 1. As the figure shows, the third operation of job 3 cannot start immediately after the second operation is completed since job 3 is transferred to machine M3 only after the first operation of job 4 is completed on machine M1. Another large idle time is on machine M1 after completing job 2: in fact, all transfers are blocked till the completion of job 3 on machine M3.

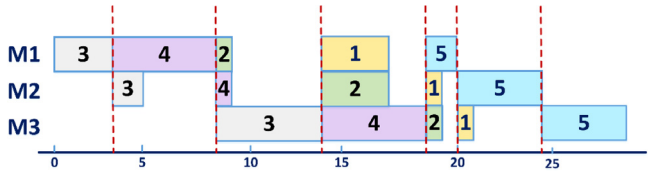


Fig. 1. Schedule for the job sequence (3, 4, 2, 1, 5) in a synchronous flow shop.

1.2. Contribution

We define a bi-objective mathematical model for the synchronous flow shop scheduling problem whose objective is to optimize the total makespan and the sum of tardiness and earliness cost, simultaneously, while accounting for synchronous material transfer.

To solve the resulting NP-hard problem, a multi-objective hybrid vibration-damping optimization (MOHVDO) algorithm is presented and compared with well-developed Pareto-based approaches, including non-dominated sorting genetic algorithm (NSGA II), multi-objective simulated annealing (MOSA), and multi-objective particle swarm optimization (MOPSO). The key feature of the proposed hybrid VDO algorithm is the integration of a neighborhood creation technique based on the imperialist competitive algorithm (ICA).

The performance of the algorithms is compared on a set of test problems using a series of measures based on non-dominant solutions, that is, Spacing metric, Diversification matrix (DM), Number of Pareto solution (NOS) metric, and CPU time. MOHVDO is shown to be the most effective solution technique.

This study is organized into seven sections. Section 2 elaborates on the state-of-the-art approaches and gaps in the literature. Section 3 presents the model description and its mathematical formalization. The solution approach is provided in Section 4. Section 5 describes the model implementation through test problem generation, parameter tuning, and performance measurements. A discussion of the outputs of the algorithms on the test problems is provided in Section 6. Finally, conclusions and future research are stated in Section 7.

2. Related works and gaps in the literature

Many researchers have focused on synchronous or blocking scheduling in the flow shop environments. Soylu et al. (2007) studied synchronous conveying among stations on a permutation flow shop-sequencing problem to minimize the makespan. Huang and Ventura (2013) proposed a flow shop scheduling problem with synchronous material movement. Waldherr and Knust (2014) investigated a modified non-preemptive permutation flow shop with synchronous movement. Kampmeyer et al. (2016) studied synchronous flow shop problems with two dominating machines. Waldherr and Knust (2016) offered decomposition algorithms for synchronous flow shop problems and extra setup times and resources. Waldherr et al. (2017) presented the consequences of considering voluntary idle times for various objective functions such as total accomplishment time, minimization of makespan, and minimizing the maximum tardiness. Wang et al. (2010) investigated blocking flow shop scheduling problems. Ribas and Companys (2015) suggest a discrete artificial bee colony algorithm to solve the blocking flow shop scheduling problem with the total flowtime criterion. Ribas and Companys (2015) developed two constructive heuristics, including HPF1 and HPF2, for the blocking flow shop problem.

Han, Gong, Li et al. (2016) suggested an improved fruit fly optimization algorithm to solve the blocking flow shop scheduling problem to minimize makespan. Han, Gong, Jin et al. (2016) formulated a multi-objective optimization problem (MOOP) for a blocking lot-streaming flow shop scheduling problem with interval processing time. Shao et al. (2017) presented the blocking flow shop scheduling problem

Table 1
A classification of related research results in the recent literature.

Study	Problem (flow shop)		Objective (Min)		Solution method									
	Synchronous	Permutation	Makespan	Earliness	Tardiness	Exact	Heuristic	GA	NSGA II	PSO	SA	HVDO	TS	Other
Companys and Mateo (2007)	✓		✓		✓	✓								
Soylu et al. (2007)	✓	✓	✓			✓								
Wang et al. (2010)	✓		✓				✓							
Bautista et al. (2012)	✓		✓		✓	✓								
Moslehi and Khorasanian (2013)	✓		✓		✓	✓								
Huang and Ventura (2013)	✓		✓			✓	✓							
Waldherr and Knust (2014)	✓	✓	✓		✓		✓							
Ribas and Companys (2015)	✓													✓
Ribas and Companys (2015)	✓						✓							
Kampmeyer et al. (2016)	✓		✓										✓	
Waldherr and Knust (2016)	✓		✓			✓								
Han, Gong, Li et al. (2016)	✓		✓											✓
Han, Gong, Jin et al. (2016)	✓		✓	✓			✓							
Shao et al. (2017)	✓				✓									✓
Waldherr et al. (2017)	✓		✓		✓	✓	✓							
Gong et al. (2018)	✓		✓	✓										✓
Schaller and Valente (2019)		✓		✓	✓	✓								
Han et al. (2020)	✓		✓				✓							✓
Hamzadayı (2020)		✓	✓			✓								
Kurdi (2020)		✓	✓					✓			✓			✓
Perez-Gonzalez et al. (2020)		✓	✓				✓							✓
Luo (2020)		✓												✓
Lang et al. (2021)		✓												✓
Nazif (2021)		✓												✓
Doush et al. (2022)		✓					✓							✓
Chen et al. (2021)		✓	✓											✓
Ribas et al. (2021)	✓	✓	✓				✓							✓
Safari et al. (2022)		✓	✓											✓
An et al. (2022)		✓												✓
Brammer et al. (2022)		✓												✓
Ferreira et al. (2022)		✓												✓
He et al. (2022)		✓												✓
Koulamas and Kyparisis (2022)	✓	✓												✓
This study	✓	✓	✓	✓	✓	✓	✓		✓		✓	✓		

to decrease the total. Gong et al. (2018) studied the blocking lot-streaming flow shop scheduling problem. Branch-and-bound algorithms were developed by Schaller and Valente (2019) to minimize total earliness and tardiness in a two-machine permutation flow shop. Miyata and Nagano (2019) presented a comprehensive review of the m-machine flow shop scheduling problem with blocking situations. An effective benders decomposition algorithm to solve the distributed permutation flow shop scheduling problem was proposed by Hamzadayı (2020). Kurdi (2020) proposed a memetic algorithm with novel semi-constructive crossover and mutation operators to minimize makespan in the permutation flow shop scheduling problem. Perez-Gonzalez et al. (2020) addressed the permutation flow shop scheduling problem with availability constraints.

A few exact methods have also been developed for blocking flow shop scheduling problems with makespan and tardiness criteria (Bautista et al., 2012; Companys & Mateo, 2007; Moslehi & Khorasanian, 2013).

Among the most recent work addressing permutation flow shop scheduling problems we find: models focusing on the sustainable aspects of blocking flow shop scheduling problems such as the one defined by Han et al. (2020), where a multi-objective optimization problem is formalized based on makespan and energy consumption criteria; models assuming that the setup time of machines depends not only on the jobs to be processed but also on the previously processed ones like the parallel flow shop configuration analyzed by Ribas et al. (2021) under the blocking constraint: models proposing hybrid techniques, such as island neighboring heuristics harmony search algorithm (INHS) developed by Doush et al. (2022); models of simultaneous minimization of the mean and variation of job waiting times within a blocking (synchronous) setting like the one of Koulamas and Kyparisis (2022); models exploiting the parallel algorithms idea to enhance the quality of the results, such as the parallel ant colony optimization algorithm developed by of Nazif (2021); models that attempt to address the problem of predicting the dynamic production status of flow shops and estimating jobs' makespan, such as the bilevel interactive optimization (BIO) approaches proposed by Chen et al. (2021) and Safari et al. (2022); models of adaptive flexible job-shop rescheduling such as the one based on real-time order acceptance and condition-based preventive maintenance developed by An et al. (2022); models where machine learning techniques are combined with domain problem reasoning for scheduling such as the guided empirical learning process introduced by Ferreira et al. (2022) to adjust the algorithm search space and enhance the dispatching rules; models of the reinforcement learning such as the one with multiple lines and demand plans presented by Brammer et al. (2022), the Neural Network approaches used by Luo (2020) and Lang et al. (2021), and the improved Q-learning algorithm developed by He et al. (2022).

Table 1 summarizes the research work described above and provides a classification of the results achieved in the recent literature on flow shop scheduling. The use of multi-objective formulations and population-based meta-heuristics solution methods is in line with those proposed in this paper. As it can be understood from the last line in Table 1, the goal of the proposed approach is to fill a considerable gap in the literature. Indeed, despite the fact that meta-heuristic solution methods are the most used ones to solve MOOPs, and that variants of the algorithms considered and compared in this paper have been largely introduced in multi-objective environments, a simultaneous implementation of these algorithms to solve flow shop scheduling problems has been so far overlooked. A concurrent implementation of different algorithms allows for a compared study able to identify the main differences among the algorithms and their effectiveness in terms of performance measures.

3. Problem description

Consider the flow shop problem with m machine ($M_1, \ldots, M_m$), and n job ($j = 1, 2, \ldots, n$), each job being characterized by m activity,

Table 2
Parameters and indices.

Indices	Notations
j	Indices for jobs where $j \in \{1, 2, \ldots, n\}$
i	Indices for machines where $i \in \{1, 2, \ldots, m\}$
k	Indices for blocks where $k \in \{1, 2, \ldots, n+m\text{-}1\}$
$p_{j,i}$	Processing time of job j on machine i
d_k	Due date of block k
C_{Ek}	Earliness cost of block k
C_{Tk}	Tardiness cost of block k

which must be done in the order, that is, $O_{j1} \rightarrow O_{j2} \rightarrow \ldots \rightarrow O_{jm}$. The O_{ji} activity must be processed continuously at time $p_{j,i}$ on the machine M_i. All jobs and machines are accessible at zero time.

All jobs must be processed in the same order on all machines, and no preemption is allowed. In one cycle, the start time of all current jobs on their machines is synchronous. In this case, all the processed jobs will stay until the last job is completed. Then all the jobs are transmitted to the following machine simultaneously. In other words, since there are no intermediate buffers, if machine $i+1$ is not empty, machine i will be blocked by a certain job j (in a certain block k). As a result, the cycle time is defined by the maximum processing time of its operation. This synchronous transfer becomes more critical when the space between stations is limited or the job's size is large.

The goal is to define the sequence of the operations so that the makespan and the total tardiness and earliness cost are minimized. The parameters and indices are outlined in Table 2.

The following variables are determined in the model:

$X_{j,k}$: Binary variable taking value 1 if job j is processed in block k, and 0 otherwise ($k = 1, 2, \ldots, n$).

F_k: Continuous variable for the completion time of block k ($k = 1, 2, \ldots, n+m-1$).

T_k: Continuous variable for the tardiness of block k ($k = 1, 2, \ldots, n+m-1$).

E_k: Continuous variable for the earliness of block k ($k = 1, 2, \ldots, n+m-1$).

The bi-objective synchronous flow shop scheduling problem can be expressed as the following:

$$Z_1 = \min C_{\max} \tag{1}$$

$$Z_2 = \min \sum_{k=1}^{n+m-1} (C_{Ek} \cdot E_k + C_{Tk} \cdot T_k) \tag{2}$$

s.t.

$$\sum_{k=1}^{n} X_{j,k} = 1 \qquad \forall j = 1, 2, \ldots, n; \tag{3}$$

$$\sum_{j=1}^{n} X_{j,k} = 1 \qquad \forall k = 1, 2, \ldots, n; \tag{4}$$

$$F_1 \geq \sum_{j=1}^{n} X_{j,1} \cdot p_{j,1} \qquad \forall j = 1, 2, \ldots, n; \tag{5}$$

$$F_k \geq F_{k-1} + \sum_{j=1}^{n} X_{j,k-i+1} \cdot p_{j,i} \qquad \forall 1 < k < n+m, \max\{1, k-n+1\} \leq i \leq \min\{k, m\}; \tag{6}$$

$$F_k - \sum_{j=1}^{n} d_k \cdot X_{j,k} = T_k - E_k \qquad \forall k = 1, 2, \ldots, n; \tag{7}$$

$$C_{\max} \geq F_{k+m-1} \tag{8}$$

$$T_k = \max\{F_k - d_k, 0\} \qquad \forall k = 1, 2, \ldots, n+m-1; \tag{9}$$

$$E_k = \max\{d_k - F_k, 0\} \qquad \forall k = 1, 2, \ldots, n+m-1; \tag{10}$$

$$T_k \geq 0 \qquad \forall k = 1, 2, \ldots, n+m-1; \tag{11}$$

$$F_k \geq 0 \qquad \forall k = 1, 2, \ldots, n+m-1; \tag{12}$$

$$X_{j,k} \in \{0, 1\} \qquad \forall j, k = 1, 2, \ldots, n \tag{13}$$

The objectives are to minimize the total makespan and total tardiness cost plus earliness, Eqs. (1) and (2), respectively. Constraints (3) and (4) confirm that each job is allocated to precisely one position, and each position is occupied by exactly one job, respectively. Constraints (5) and (6) state that the completion times of blocks are determined. Constraint (8) indicates the domains of makespan.

Constraint (7) calculates the tardiness and earliness in block k using the accomplishment time on the machine m and the delivery time d. If the work is delayed in position k, the left side of the Constraint (7) is strictly positive ($T_k > 0, E_k = 0$), and if the work in position k is processed early, the left side of the Constraint (7) is strictly negative ($T_k = 0, E_k > 0$). When the work in position k is on time, the left side will be zero ($T_k = E_k = 0$). Constraints (9) and (10) represent the tardiness and earliness of the blocks, respectively. Finally, constraints (11) to (13) describe the domains of the variables.

4. Solution approach

4.1. Multi-objective optimization problem (MOOP)

The MOOP refers to problems with two or more goals that must be satisfied simultaneously. These kinds of problems do not normally admit a unique solution but a group of solutions named Pareto's optimal or non-dominant solutions. The general MOOP model is defined as follows (Ahmed & Deb, 2013):

$$\begin{aligned} &Min \;\; y = f(x) = [f_1(x), \; f_2(x), \ldots, \; f_L(x)] \\ &\text{s.t.} \\ &g_s(x) \leq 0, \forall s = 1, 2, \ldots, \; S; \\ &h_t(x) = 0, \forall t = 1, 2, \ldots, \; T. \end{aligned} \tag{14}$$

Here x is the vector of decision variables, $f(x)$ represents the vector of objective functions, and $(g_s(x))_{s=1,\ldots,S}$ and $(h_t(x))_{t=1,\ldots,T}$ are the vectors of constraints. In the vector of the objective functions $f(x)$, often, a few functions conflict with each other, with some functions being Min objectives and others Max objectives.

In a minimization problem for all objectives, x_1 dominates $x_2 (x_1 \succ x_2)$ if and only if two of the following situations are true:

a) For the whole objectives, x_1 is not worse than x_2, that is: $f_l(x_1) \leq f_l(x_2), \forall l = 1, 2, \ldots, \; L$.
b) For at least one objective, x_1 is strongly better than x_2, that is: $f_l(x_1) < f_l(x_2), \exists l = 1, 2, \ldots, \; L$.

Pareto optimal solutions are employed in the Pareto set in the archive. The archive solutions can be updated with more appropriate solutions with the repeat process.

Academics have established various methods to solve the MOOP (Amin-Tahmasbi & Tavakkoli-Moghaddam, 2011; Khalili & Tavakoli-Moghadam, 2012; Zeng et al., 2013), which are generally divided into five categories: (1) Scalar methods; (2) Interactive methods; (3) Fuzzy methods; (4) Meta-heuristic methods; (5) Decision methods. Among these categories, the meta-heuristic methods are the ones that have been used more (Collette & Siarry, 2003). Among these methods, the population-based ones have been frequently applied in the literature showing their general superior performances (Pasandideh et al., 2011). In particular, NSGA II algorithms (Reddy et al., 2017), simulated annealing (SA) (Pasandideh et al., 2013), particle swarm optimization (PSO) (Tsai et al., 2014), and vibration damping optimization (VDO) (Mehdizadeh et al., 2016) have been used and compared in multi-objective environments. Following this line of thought, in this study, we solve the proposed model using an epsilon-constraint method and multi-objective meta-heuristic algorithms (i.e., NSGA II, MOSA, MOPSO, and MOHVDO).

The rest of this section is dedicated to introducing how we define the chromosome, crossover, and mutation operators for implementing the meta-heuristic algorithms and describing the meta-heuristics employed.

Table 3
The processing time assignment of each job (Jj) to each machine (Mi).

	M1	M2	M3	M4	M5
J1	0.8003	0.0357	0.6555	0.8235	0.7655
J2	0.1419	0.8491	0.1712	0.6948	0.7952
J3	0.4218	0.9340	0.7060	0.3171	0.1869
J4	0.9157	0.6787	0.0318	0.9502	0.4898
J5	0.7922	0.7577	0.2769	0.0344	0.4456
J6	0.9595	0.7431	0.0462	0.4387	0.6463
J7	0.6557	0.3922	0.0971	0.3816	0.7094

4.2. Chromosome, crossover, and mutation

In our model, the chromosomes have three parts: the sequence of jobs, the assignment of a job to the machine, and the machine's allocation to the block. To begin with, we want to know which job should be processed first. For example, consider the situation where there are seven jobs, five machines, and six blocks. Create a randomized matrix (array) of length seven (as the number of jobs) between 0 and 1 as follows:

0.5469	0.9575	0.9649	0.1576	0.9706	0.9572	0.4854

Then write the generated matrix in descending order based on their priority number as follows:

4	7	1	6	2	3	5

In this first stage, the sequence according to which the jobs should be processed is determined.

The second part of the chromosome allows to know which job should be assigned to each machine. To do this, we need to create a matrix with seven rows (jobs) and five columns (machines). Table 3 shows an example of randomized matrix accounting for the processing time assignments of seven jobs to five machines. In this matrix, we select the largest number for each row, thus assigning job 1 to machine 4, job 2 to machine 2, and so on until the end.

The third part of the chromosome yields each machine's assignment to each block and is characterized by a matrix with five rows (machines) and six columns (blocks). Table 4 shows an example of randomized matrix for the assignment of five machines to six blocks. This matrix selects the largest number in each row, indicating each machine's allocation to a block. As a result, machine 1 is assigned to block 6, machine 2 to block 4, and so on until the end.

Regarding the other operators, crossover operators are single-point and two-point. Mutation operators are uniform, swap, and reversion.

A penalty policy is employed to control infeasible solutions. In the proposed algorithm, the penalty is specified as a positive and known coefficient of constraint violation. The larger is the coefficient, the larger is the penalty. If a chromosome is feasible, its penalty is zero. Finally, a chromosome's fitness function is described as the total of its objective function and penalty (Pasandideh & Niaki, 2012).

4.3. Multi-objective hybrid vibration damping optimization (MOHVDO) algorithm

Mehdizadeh et al. (2015) introduced VDO for single-objective optimization problems based on the idea of vibration damping in mechanical vibration. Then, Hajipour et al. (2014) introduced multi-objective VDO (MOVDO) for solving the MOOP.

In this study, we propose a multi-objective hybrid VDO (MOHVDO). The key and new feature of this algorithm compared to MOVDO is the incorporation of a neighborhood creation technique based on the imperialist competitive algorithm (ICA) (Nabipoor Afruzi et al., 2014).

The implementation of neighborhood creation in the MOVDO algorithm has already been considered in previous research works. See Mehdizadeh et al. (2016) and Nobari et al. (2018), among others. However, to the best our knowledge, none of the techniques employed so far was developed using the ICA approach. As we will show later, the performance of the proposed algorithm is enhanced with a hybridization of MOVDO with ICA.

Table 4
The assignment of each machine (Mi) to each block (Bk).

	B1	B2	B3	B4	B5	B6
M1	0.7547	0.119	0.2238	0.8909	0.2575	0.9293
M2	0.2760	0.4984	0.7513	0.9593	0.8407	0.3500
M3	0.6797	0.9597	0.2551	0.5472	0.2543	0.1966
M4	0.6551	0.3404	0.506	0.1386	0.8143	0.2511
M5	0.1626	0.5853	0.6991	0.1493	0.2435	0.6160

The neighborhood process in ICA is conducted by producing a random variable (x) that operates as uniform distribution, that is, $x \sim Uniform(0, \beta \times d); \beta > 1$. Parameter d designates the distance linking a colony and an imperialist. Another characteristic specified in the literature of ICA to search for variations across an imperialist is the deviation of a path (θ). This is introduced by defining a uniform distribution $\theta \sim U(-\gamma, \gamma)$. Parameter γ is described as a deviation from the original direction (for details, please see Alaghebandha and Hajipour (2013)).

The VDO algorithm commences by determining population size (*nPop*), preliminary amplitude (A_0), the greatest number of iterations at each amplitude (L), standard deviation (σ), damping coefficient (γ), and generating random solutions in the search space. These solutions are appraised through the objective function value (E). Hence, comparing, sorting, and ranking of the current population (P) is performed using the fast non-dominated sorting (FNDS) and the crowding distance (CD) function of the NSGA II algorithm. As a result, all the non-dominant chromosomes of the first iteration are found. The primary loop produces a solution randomly, and then, utilizing neighborhood structure, a fresh solution in the new population (S) is obtained, and the best one is selected. This population is evaluated and sorted using FNDS and CD, selecting the best ones.

The new solution is admitted if (Mehdizadeh et al., 2015):

$$\Delta = E(\text{new solution}) - E(\text{current solution}) < 0 \tag{15}$$

If $\Delta > 0$, a random number r between (0, 1) is produced. The present solution is picked on the following criterion (Mehdizadeh et al., 2015):

$$r < 1 - \exp\left(-\frac{A^2}{2\sigma^2}\right) \tag{16}$$

So, the results are compared using Eq. (16) and the CD operator. Suppose that two solutions, x and y, are considered in the population. If $r_x < r_y$ or (if $r_x = r_y$ and $d_x < d_y$), where d_x and d_y are CDs, then $x \succ y$. After this step, the second loop that regulates the amplitude is applied for dropping amplitude at each iteration. The algorithm is finished once the following stopping criterion is met (Mehdizadeh et al., 2015):

$$A_t = A_0 \exp\left(-\frac{t}{Q}\right) \tag{17}$$

If the algorithm is not stopped, a new population (Q) is created using the tournament operator. Through the elitism process, two populations P and Q are combined and produce a new population $R = P \cup Q$. The population R is sorted by implementing FNDS and CD. Afterward, the following iteration population, as big as *popsize* is selected. Finally, a group of the next iteration population *P+1* is designated to have a prearranged size. The next iteration happens consistently with the last population P obtained, and the primary and secondary loops are carried on for all the following iterations until the last one. The complete pseudo-code of the MOHVDO is shown in Fig. 2.

***Step 1*:** Initializing the algorithm parameters, which consist of:
population size (nPop),
initial amplitude (A_0),
maximum number of iterations at each amplitude (L),
damping coefficient (γ),
standard deviation (σ),
$Q = {}^{2}/_{\gamma}$
***Step 2*:** Generating a feasible initial solution and t is set at one (t=1)
***Step 3*:** Calculating the objective value E_0 for the initial solution.
Perform fast non-dominate sorting (FNDS) and calculate ranks
Calculate crowding distance (CD)
Sort population according to ranks and CDs
***Step 4*:** Initializing the internal loop. In this step, the internal loop is carried out for l =1 and repeat while l < L.
Step 5*:** Neighborhood generation: ***use ICA method and calculate new objectives (E).
***Step 6*:** Accepting the new solution.
Set $\Delta = E - E_0$.
If $\Delta < 0$, accept the new solution, else if $\Delta > 0$ generate a random number r between (0, 1);
If $r < 1 - \exp(-\frac{A^2}{2\sigma^2})$, then accept a new solution; otherwise, reject the new solution and accept the previous solution.
***Step 7*:** If $l > L$, then update A ($A_t = A_0 \exp(-\frac{t}{Q})$) and $t+1 \rightarrow t$; otherwise l +$l \rightarrow l$ and go back to Step 5. If $A_t = 0$ return to Step 8; otherwise, go back to Step 4.
***Step 8*:** Stopping criteria. In this step, the proposed algorithm will be stopped after pre-specified maximum iteration is achieved.
Q = New Population
R = P ∪ Q
Perform fast non-dominate sorting (FNDS) on R and calculate ranks
Calculate crowding distance (CD) of R
Sort population according to ranks and CDs on R
Create P.+1 as Size as Population Size (Population= P.+1)
At the end, the best solution is obtained.

Fig. 2. Pseudo-code of MOHVDO algorithm.

4.4. Non-dominated Sorting Genetic Algorithm II (NSGA II)

Non-dominated Sorting Genetic Algorithm (NSGA II) was introduced by Srinivas and Deb (1994) and has been developed up to 2000 (Collette & Siarry, 2003). To solve the MOOP, NSGA II is one of the most effective evolutionary algorithms (Khalkhali et al., 2016). In place of an enhanced GA, the NSGA II solution is comparable to the common GA (Liu et al., 2019). The main concept of NSGA II is to apply both CD and the non-dominated sorting (NDS) algorithm to compute the individual's fitness value in a population (Zitzler & Thiele, 1998). After that, the following population can be produced based on all the individuals' achieved fitness values. The correct fitness values are very significant for the quick convergence of the NSGA II algorithm (Liu et al., 2019). For more details about the algorithm steps and NSGA II pseudo-code process, see Moslemi et al. (2017).

4.5. Multi-objective simulated annealing (MOSA)

The heuristic technique entitled "Simulated Annealing" (Kirkpatrick et al., 1983) was initially considered for solving the single-objective optimization problem and soon confirmed to be convergent and robust if the annealing is adequately and gradually performed (Mitra et al., 1986). The multi-objective simulated annealing (MOSA) applies the domination idea and the annealing structure for well-organized exploration (Nam & Park, 2000). MOSA approaches that integrate the Pareto set have also been developed quite soon after its introduction (Czyzak & Jaszkiewicz, 1998; Nam & Park, 2000; Suppapitnarm et al., 2000; Ulungu et al., 1999). The acceptance criteria of these methods are all based on the degree of difference between the new and current solutions. Overall, the MOSA should produce Pareto optimal solutions with a fine level of variation (Zidi et al., 2012). In this research, we follow the method of Zidi et al. (2012).

4.6. Multi-objective particle swarm optimization (MOPSO)

The multi-objective particle swarm optimization (MOPSO) algorithm, primarily suggested by Coello et al. (2004), is derived from the traditional PSO. MOPSO needs to combine Pareto dominance and PSO to solve the MOOP. In addition, this algorithm employs a constraint-handling procedure and a distinctive mutation operator that significantly enhances the exploratory capabilities. For more information about the main steps of this algorithm, see Khalkhali et al. (2016).

4.7. Epsilon-constraint method

Optimization is to find one or more feasible solutions consistent with the ultimate values of one or more goals. Finding such optimal points can be related to minimizing costs or maximizing profits. One common method for obtaining an effective boundary is to add a constraint or epsilon-constraint.

Table 5
Ten different problems for 12 different combinations.

Problem size	Small		Medium		Large	
Machine	5	10	15	20	30	40
Job	50 100	50 100	200 300	200 300	400 500	400 500

In this technique, one of the objective functions is considered the key objective function, while all the other objective functions are assigned an ε limit. The epsilon-constraint method can be used to obtain the Pareto frontier. The limit of ε may differ for different objectives for the Pareto solution set. The following model is used to find the optimal points (Narendra & Liang, 2008):

$$\begin{aligned} &Min\, f_1(x) \\ &\text{s.t.} \\ &x \in X; \\ &f_2(x) \le \varepsilon_2; \\ &. \\ &. \\ &f_L(x) \le \varepsilon_L. \end{aligned} \tag{18}$$

5. Implementation of the model: Test problems, performance measures, fine-tuning

The implementation of the model and algorithms is similar to the one presented by Waldherr and Knust (2016). For the ease of reading, this section is organized in four subsections each one of them addressing the main phases that the implementation comprises.

After describing the test problems, the metrics used to compare the performance of the algorithms are introduced. Hence, the computational performance of the algorithms on the set of test problems is analyzed. The parameters of the proposed algorithms are tuned according to the Taguchi method.

5.1. Test problem generation

We considered random examples for three sizes of problems, i.e., small size, medium size, and large size problems. We generated twelve different test problems obtained by considering combinations of 5, 10, 15, 20, 30, and 40 machines and 50, 100, 200, 300, 400, and 500 jobs. Table 5 shows the 12 combinations of machine and jobs for which random data were generated. The data generated for each test problem are randomized arrays and randomized matrices similar to those used in Section 4.2 as an example to explain the composition of the chromosomes. That is, for each problem an array of importance weights of the jobs was generated as well as two assignment matrices similar to those reported in Tables 3 and 4. For instance, keeping in mind that the maximum number of blocks is $n+m-1$, with n = number of jobs and m = number of machines (see Table 2), in the case of the small size problem with 5 machines and 50 jobs, the two assignment matrices had dimension 50 × 5 (jobs × machines) and 5 × 54 (machine × blocks). The assignment matrices elaborated for the medium size problem with 15 machines and 200 jobs had dimension 200 × 15 (jobs × machines) and 15 × 214 (machine × blocks), while those of the large size problem of 30 machines and 400 jobs were of dimension 400 × 30 (jobs × machines) and 30 × 429 (machine × blocks).

Each problem was executed 10 times, and their averages used for the calculation and comparison of the algorithms. The numerical testing was performed on a laptop with a Core i7 (2.5 GHz) processor and one GB of RAM. The algorithms were coded in MATLAB (Version R2016a).

5.2. Performances measures

Since multi-objective problems produce non-dominant solutions, for comparing algorithms, we need criteria that are designed based on non-dominant solutions (Coello Coello et al., 2002). In this paper, the following performance metrics are used for the comparison.

I. *Spacing metric* The spacing metric allows us to determine the uniformity of the point spread inside the solution set. The definition of this metric is given below:

$$S = \frac{\sum_{i=1}^{N-1} \left| d_i - \overline{d} \right|}{(N-1)\cdot \overline{d}},\ \text{with } d_i = \min_{\substack{k=1,\dots,N \\ k\neq i}} \sum_{j=1}^{L} |f_j^i - f_j^k| \text{ and } \overline{d} = \sum_{i=1}^{N} \frac{d_i}{N} \tag{19}$$

Here f_l^i ($l = 1, \dots, L$ and $i = 1, \dots, N$) are the solutions in the set of non-dominant solutions to objective function $f_l(x)$. L is the number of objectives.

II. *Diversification Matrix (DM)* This performance criterion provides a clue of the diversity of solutions gotten from a given algorithm and is computed by Eq. (20):

$$D = \sqrt{\sum_{l=1}^{L} \left(\max_i f_l^i - \min_i f_l^i \right)^2} \tag{20}$$

In this criterion, the length of the diameter of the cubic space is calculated using the most distant values of non-dominant solutions. L is the number of objectives. This distance is obtained by using the Euclidean distance between two solutions in the goal space. A larger DM value signifies a superior performance of the corresponding algorithm.

III. *Number of Pareto solution (NOS) metrics*

This performance measure counts the total number of non-dominated solutions generated by the compared algorithms. A larger NOS value indicates a superior performance of the corresponding algorithm.

IV. *CPU time (or processing time)* This is the amount of time for which a central processing unit (CPU) was used for processing instructions of a computer program or operating system. The CPU time is measured in clock ticks or seconds and it includes only the time during which the program actually uses the CPU to perform tasks such as doing arithmetic and logic operations. For example, waiting for input/output (I/O) operations or entering low-power (idle) mode are not CPU time.

5.3. Meta-heuristics calibration

Because all meta-heuristic algorithms depend highly on their parameters, the Taguchi method (Afzalirad & Rezaeian, 2016) has been performed to improve the performance of the algorithms. As the smaller-the-better rule categorizes objective functions in our model, it is consistent with using Signal-To-Noise Ratio (SNR) calculations:

$$SNR = -10\log_{10}(objective\ function\ or\ mean\ of\ the\ Performance\ Measurement)^2 \tag{21}$$

Furthermore, the formula computing the relative deviation index (RDI) is given by:

$$RDI = \frac{OB_x - OB_{best}}{OB_{worst} - OB_{best}} \times 100 \tag{22}$$

where OB_x is the value of the performance measure obtained by each algorithm for a given problem and OB_{best} is the best (OB_{worst} is the worst) value of the performance measure gained by algorithms for the associated problem.

To execute the Taguchi method, all parameters' levels must be defined. Table 6 shows the levels of all the parameters defined for the simulations with the test problems using MOHVDO, NSGA II, MOSA, and MOPSO. In connection with the outputs of MINITAB software, optimal values of the algorithm's parameters, along with the maximum SNR and minimum RDI rules (Alaghebandha & Hajipour, 2013; Choi & Kim, 2009), are determined.

Table 6
Parameters levels in MOHVDO, NSGA II, MOSA, and MOPSO.

	Factor	Symbol	Problem	Level (1)	Level (2)	Level (3)
MOHVDO	β	A	Small	1	1	1.1
			Medium	1	1.1	1.15
			Large	1.1	1.15	1.25
	A_0	B	Small	3	5	7
			Medium	5	7	9
			Large	6	8	10
	L	C	Small	30	40	50
			Medium	40	50	60
			Large	30	40	50
	σ	D	Small	1	1.1	1.2
			Medium	1.2	1.3	1.4
			Large	1.4	1.5	1.5
	γ	E	Small	0.04	0.05	0.05
			Medium	0.7	0.9	1
			Large	1	1.3	1.5
NSGA II	*Pc*	A	Small	0.5	0.6	0.7
			Medium	0.6	0.7	0.8
			Large	0.7	0.8	0.9
	Pm	B	Small	0.1	0.2	0.3
			Medium	0.2	0.3	0.4
			Large	0.2	0.3	0.4
	Pop_size	C	Small	40	70	100
			Medium	60	70	80
			Large	80	90	100
	Max_iteration	D	Small	30	40	50
			Medium	50	70	90
			Large	100	200	250
MOSA	*Initial temperature* (T_0)	A	Small	200	300	400
			Medium	500	600	700
			Large	800	900	1000
	Temperature reduction multiplier	B	Small	0.7	0.8	0.9
			Medium	0.75	0.85	0.95
			Large	0.9	0.95	0.99
	Pop_size	C	Small	20	30	40
			Medium	30	40	50
			Large	35	45	55
	Max_iteration	D	Small	10	15	20
			Medium	20	25	30
			Large	40	50	60
MOPSO	*number of particle*	A	Small	100	130	160
			Medium	140	150	170
			Large	150	170	190
	Cognitive factor	B	Small	0.5	0.6	0.7
			Medium	0.7	0.8	0.9
			Large	0.8	0.9	1
	Social factor	C	Small	0.5	0.6	0.7
			Medium	0.7	0.8	0.9
			Large	0.8	0.9	1
	Max_iteration	D	Small	20	30	40
			Medium	30	40	50
			Large	40	50	60

5.4. Algorithm performance outputs

The algorithms' outputs are compared with each other according to performance measurements. Table 7 reports the performance outputs of the algorithms for all the criteria and for all the test problems.

Fig. 3 provides a graphical comparison of the NOS evaluated for different test problems and different algorithms. We use the Tukey statistical analysis method at a 95% confidence level and Relative percentage deviation (RPD) to compare the algorithms. RPD is obtained as Eq. (23)

$$RPD = [(MIN_{stage} - MIN_{total})/MIN_{total}] \times 100\%. \quad (23)$$

Fig. 4 provides a comparison of the performance of all the algorithms. The MOHVDO algorithm is better than the other algorithms concerning all the performance measures except the CPU TIME index. That is, the MOHVDO algorithm gives higher NOS and MD, lower spacing, and Min objectives. Finally, MOSA shows the best results in terms of CPU TIME index.

6. Discussion

Considering that the MOHVDO algorithm and MOSA algorithm show the best performance with respect to the other algorithms in terms of objective functions and CPU Time, respectively, they have been compared with the epsilon-constraint method. The comparison of the results obtained from applying MOHVDO and the epsilon-constraint method and that of the results from MOSA and the epsilon-constraint method are described in Table 8. On average, the problem-solving time through the epsilon-constraint method is 3021.31 s. This value is 12.52 s for MOSA, representing a very good time for a meta-heuristic

Table 7
Computational result of the algorithms for all problems.

Size	Problem	Job	Machine	Objective 1				Objective 2			
				MOPSO	NSGA II	MOHVDO	MOSA	MOPSO	NSGA II	MOHVDO	MOSA
Small	1	5	50	351	356	314	388	914	912	694	1162
	2	5	100	785	834	750	834	1414	2221	2147	2852
	3	10	50	184	222	207	261	572	896	674	966
	4	10	100	409	403	363	495	1626	2188	1620	2286
Medium	5	15	200	655	643	578	818	4236	6397	4579	5230
	6	15	300	965	1029	945	1198	7916	9793	8346	9427
	7	20	200	704	664	705	928	6465	7440	4072	8600
	8	20	300	769	754	831	1013	10980	10714	8029	9676
Large	9	30	400	764	761	695	977	12978	15605	9641	16517
	10	30	500	918	942	877	1122	16389	19735	10728	22263
	11	40	400	784	697	618	851	11939	12616	10680	16975
	12	40	500	807	752	768	943	14754	18457	10412	21107
Size	Problem	Job	Machine	CPU Time				Spacing			
				MOPSO	NSGA II	MOHVDO	MOSA	MOPSO	NSGA II	MOHVDO	MOSA
Small	1	5	50	12.19	11.17	12.28	6.39	0.8922	0.9591	1.2711	0.9025
	2	5	100	15.33	13.68	15.31	7.04	0.8552	0.8441	0.7096	1.4178
	3	10	50	13.31	17.65	16.66	6.55	0.9195	0.8797	0.8403	0.9527
	4	10	100	14.04	13.16	15.21	7.19	1.2599	0.7099	0.7651	1.0231
Medium	5	15	200	18.38	19.88	22.31	23.03	1.1051	0.7837	1.0222	0.8431
	6	15	300	22.97	25.66	28.96	15.69	0.8742	0.8625	0.6574	1.1076
	7	20	200	24.61	28.42	30.27	16.82	0.9916	1.0048	0.8205	0.8877
	8	20	300	23.45	28.15	30.43	17.42	1.055	0.8969	0.954	1.0694
Large	9	30	400	29.93	36.59	42.25	23.21	0.9486	1.1533	0.8313	1.1782
	10	30	500	35.26	45.06	49.95	27.62	0.9389	1.3299	1.018	1.0939
	11	40	400	40.61	50.96	55.53	34.48	0.4741	1.0503	0.9443	1.015
	12	40	500	38.32	48.68	53.15	34.41	0.9681	1.0663	1.0243	0.9474
Size	Problem	Job	Machine	NOS				MD			
				MOPSO	NSGA II	MOHVDO	MOSA	MOPSO	NSGA II	MOHVDO	MOSA
Small	1	5	50	7	7	8	9	25.658	23.252	23.821	46.827
	2	5	100	9	8	6	5	38.967	39.48	17.843	50.047
	3	10	50	7	6	7	9	32.849	24.242	23.959	46.505
	4	10	100	6	5	5	5	25.513	18.726	18.787	34.984
Medium	5	15	200	13	6	7	6	84.229	50.701	44.783	70.057
	6	15	300	5	10	4	6	65.651	96.696	29.578	96.82
	7	20	200	12	8	5	10	89.463	42.079	48.268	141.14
	8	20	300	6	16	6	9	53.838	135.96	45.118	144.8
Large	9	30	400	8	13	5	14	90.81	112.91	53.477	226.92
	10	30	500	9	9	10	10	103.67	86.47	107.8	224.35
	11	40	400	3	11	8	14	16.816	88.901	76.773	222.86
	12	40	500	6	10	8	10	88.266	114.19	109.64	203.96

Table 8
Comparison of MOHVDO and MOSA algorithms with the epsilon-constraint method.

Size	Problem	Epsilon-constraint method			MOHVDO			MOSA			Error (%) MOHVDO	
		Obj 1	Obj 2	Time	Obj 1	Obj 2	Time	Obj 1	Obj 2	Time	Obj 1	Obj 2
Small	1	314	694	134.19	314	694	12.28	388	1162	6.39	0	0
	2	750	2147	295.68	750	2147	15.31	834	2852	7.04	0	0
	3	206	672	602.59	207	674	16.66	261	966	6.55	0.48	0.29
	4	359	1606	1078.12	363	1620	15.21	495	2286	7.19	1.11	0.87
Medium	5	571	4535	4559.94	578	4579	22.31	818	5230	23.03	1.22	0.96
	6	938	8291	4597.17	945	8346	28.96	1198	9427	15.69	0.74	0.66
	7	701	4038	5533.78	705	4072	30.27	928	8600	16.82	0.57	0.84
	8	823	7963	7368.66	831	8029	30.43	1013	9676	17.42	0.97	0.82
Average		582.7	3743.2	3021.31	586.62	3770.1	23.41	741.8	5024.7	12.52	0.64	0.56

algorithm. The solution quality in the MOHVDO algorithm is acceptable regarding the best answer for each objective function. The mean error for objective functions 1 and 2 using this algorithm is 0.64% and 0.56%, respectively.

Fig. 5 shows interval plots of the simultaneous confidence intervals for Tukey's pairwise differences between algorithms. We have used MINITAB software to display these interval plots. The grouping information reported in Table 9 is based on the confidence intervals displayed in the interval plots and reveals groups of factor level means that are not significantly different. In Table 9, groups are listed in descending order of their fitted means for performance indices. Groups that share a letter, such as MOHVDO and MOPSO in MIN objective 1 are not significantly different (NSGA II vs. MOPSO). Conversely, groups that do not share a letter have very different means. Therefore, MOSA differs considerably from MOHVDO (NSGA II vs. MOHVDO, MOSA vs. MOPSO, and NSGA II vs. MOSA).

Here, we examine the interval plots to assess the simultaneous confidence intervals. If an interval does not include zero, the corresponding means are meaningfully dissimilar. The six confidence intervals in each plot of Fig. 5 compare all possible combinations of algorithms. Because

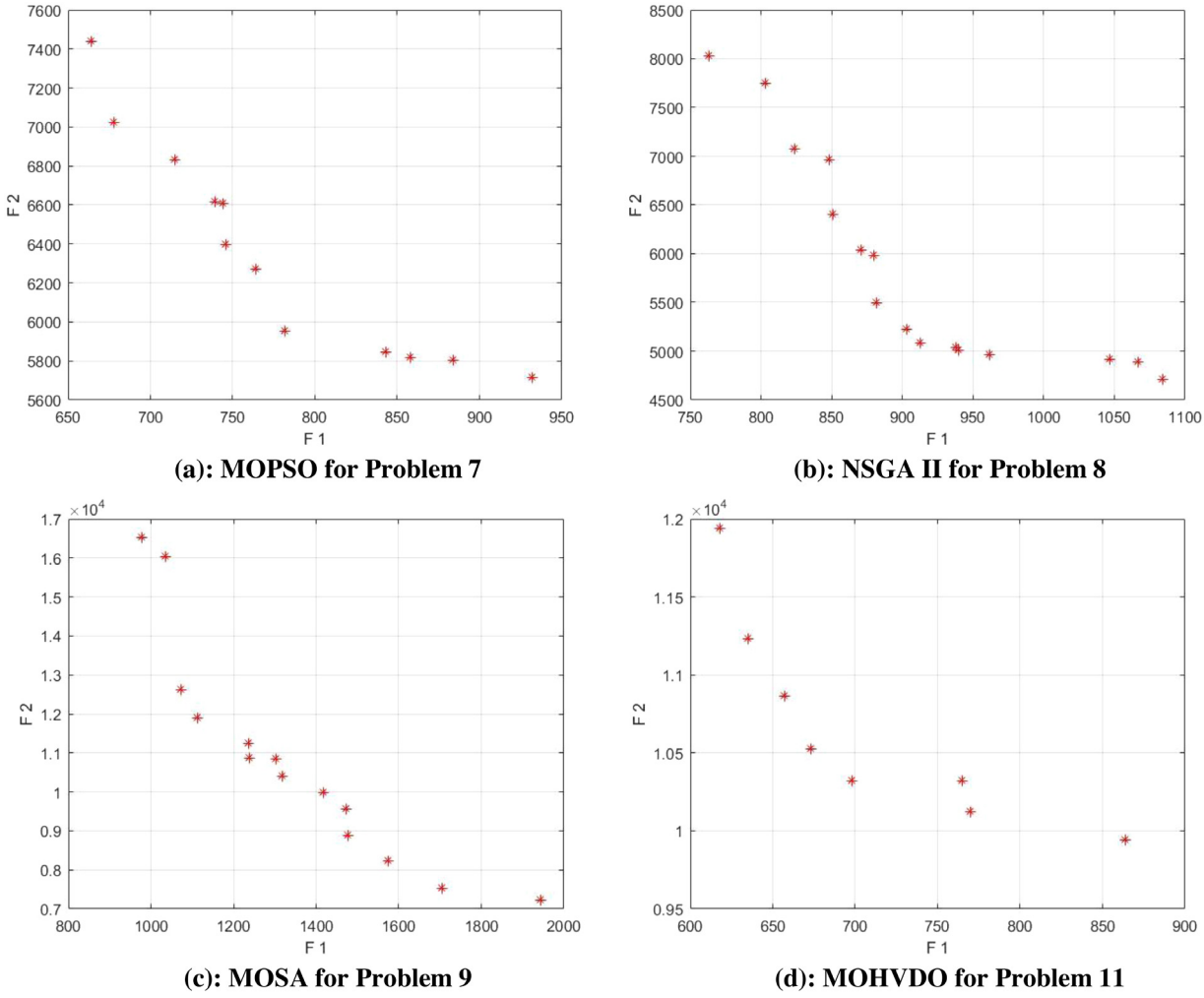


Fig. 3. Example of NOS for different test problems and algorithms.

Table 9
Tukey pairwise comparisons for performance indices.

Objective 1				**Objective 2**			
Algorithms	*N*	*Mean*	*Grouping*	*Algorithms*	*N*	*Mean*	*Grouping*
MOSA	12	0.9912	A	MOSA	12	1.3969	A
NSGA-II	12	0.6925	B	NSGA-II	12	1.1826	B
MOPSO	12	0.6703	B,C	MOPSO	12	0.7496	B
MOHVDO	12	0.5799	C	MOHVDO	12	0.6186	B
Time				**Spacing**			
Algorithms	*N*	*Mean*	*Grouping*	*Algorithms*	*N*	*Mean*	*Grouping*
MOHVDO	12	1.0913	A	MOSA	12	0.7441	A
NSGA-II	12	0.9191	B	NSGA-II	12	0.6566	A
MOPSO	12	0.7079	A,B	MOPSO	12	0.5567	A
MOSA	12	0.3310	B	MOHVDO	12	0.5300	A
NOS				**MD**			
Algorithms	*N*	*Mean*	*Grouping*	*Algorithms*	*N*	*Mean*	*Grouping*
MOHVDO	12	0.4601	A	MOHVDO	12	0.6352	A
NSGA-II	12	0.3892	A	NSGA-II	12	0.5164	A
MOPSO	12	0.2949	A	MOPSO	12	0.4889	A
MOSA	12	0.2935	A	MOSA	12	0.1199	B

these are simultaneous confidence intervals, we have 95% confidence that all the intervals include the true difference.

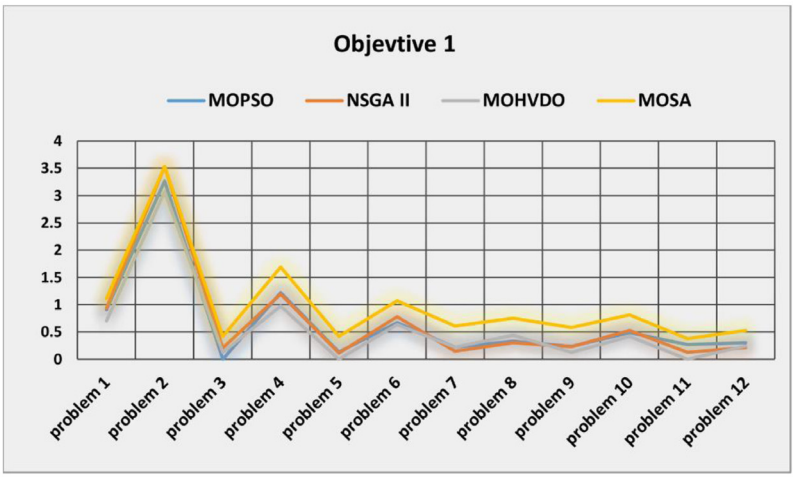
A benefit of confidence intervals is that they make the scale of the changes among the means understandable. For example, the first interval in the interval plot of MIN Objective 1, i.e., MOPSO-MOHVDO, is a confidence interval that contains zero. The same is true for the confidence interval of NSGA II vs. MOPSO. Thus, there is no evidence at $\alpha = 0.05$ for a difference in the mean. However, all other pairs of means (MOSA vs. MOHVDO, NSGA II vs. MOHVDO, MOSA vs. MOPSO, and NSGA II vs. MOSA) are significantly different because the confidence intervals do not enclose zero. Analogous considerations apply to all the other interval plots represented in Fig. 5.

7. Conclusions and recommendations for future research

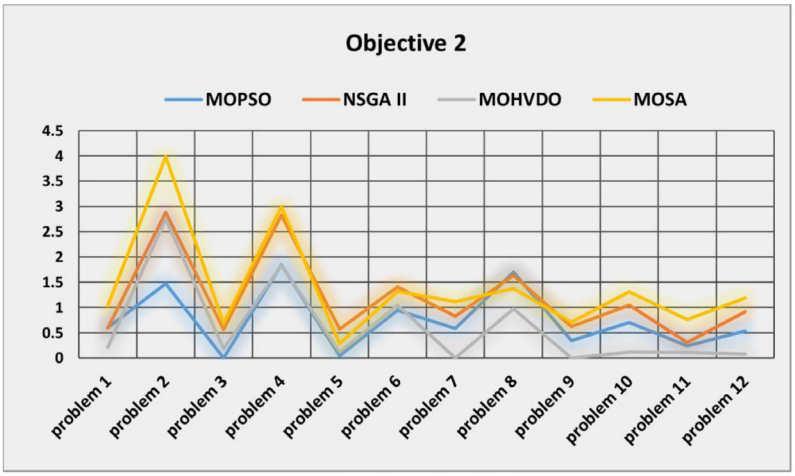
This study developed a bi-objective mathematical model for the synchronous flow shop scheduling problem. The objectives were to decrease the total makespan and sum of tardiness and earliness costs. The completion times of blocks were assumed to be determined to make the model more applicable.

Four algorithms, including NSGA II, MOSA, MOPSO, and MOHVDO, were used to find a near-optimal solution for the proposed NP-hard problem. Among these algorithms, the MOHVDO was utilized to solve the problem because of the proposed mathematical model's complexity. More precisely, we proposed a hybrid variant of MOVDO based on ICA and the integration of the neighborhood creation technique in the MOVDO algorithm.

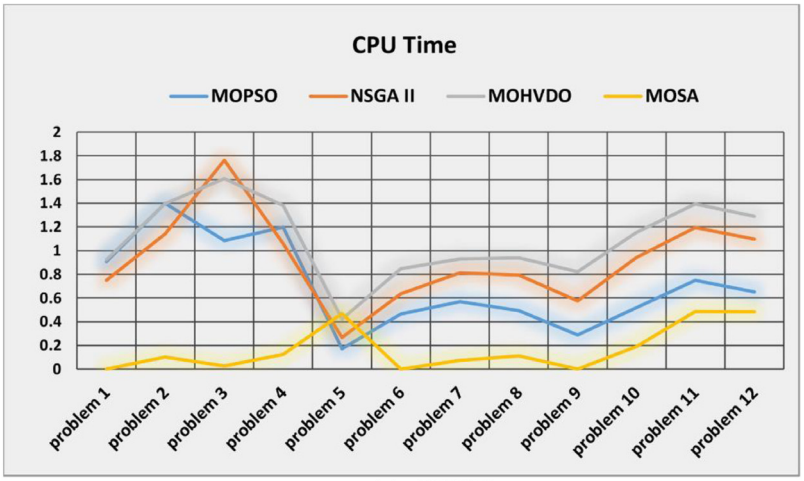
The algorithms' performance was evaluated using different measures comprising diversification, number of Pareto optimal solutions, spacing, and CPU time. The most effective solution technique resulted in MOHVDO. Furthermore, interval plots and grouping information tables were applied to determine the differences among algorithms.



(a): Objective 1



(b): Objective 2

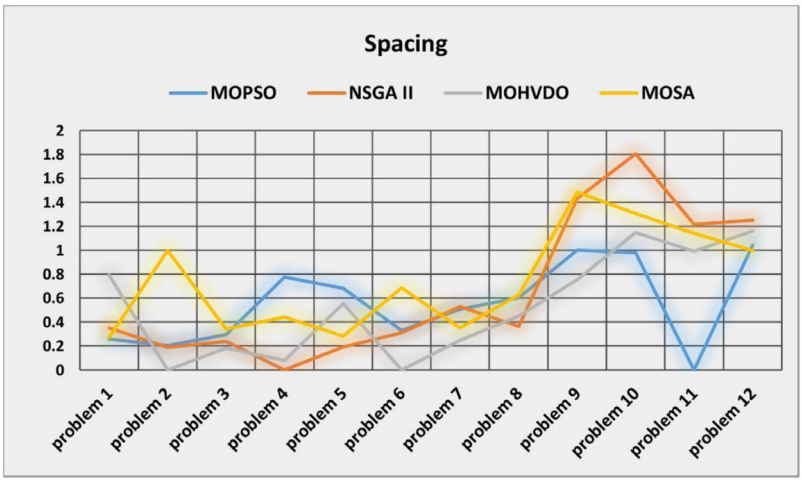


(c): CPU Time

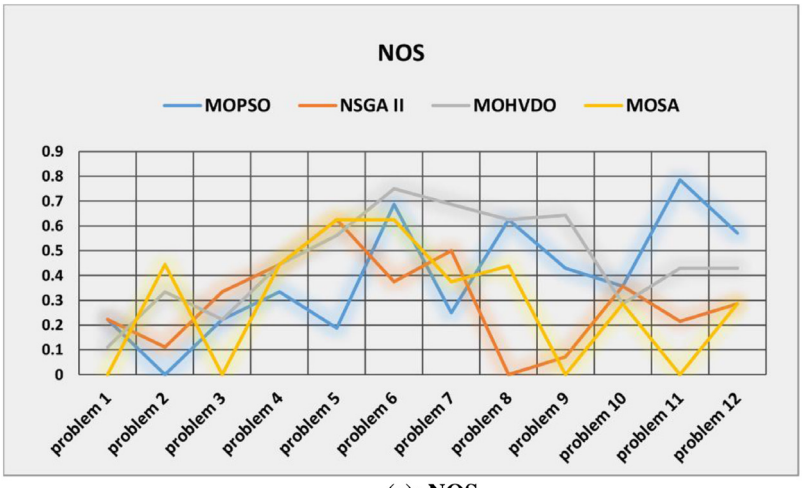
Fig. 4. Graphical comparison among the algorithms: performance indices for all test problems.

Finally, the best algorithm with respect to each performance measure was identified.

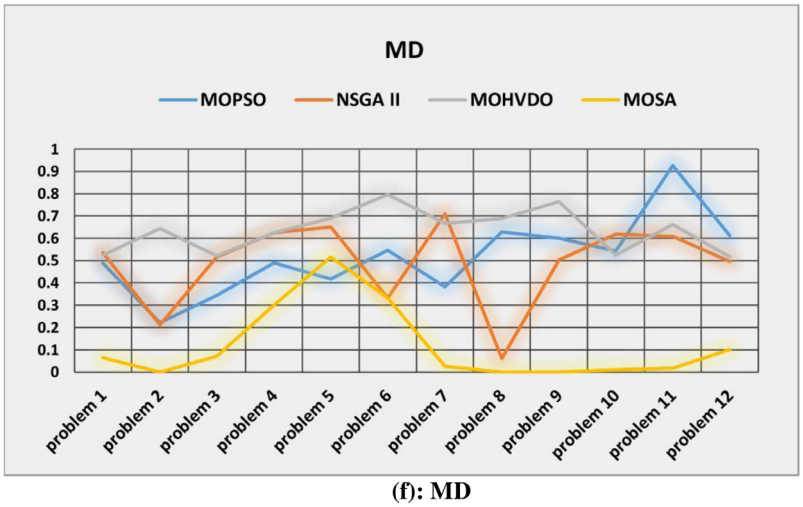
MOHVDO and MOSA showed the best performance with respect to the other algorithms in terms of objective functions and CPU Time, respectively, and were compared with the epsilon-constraint method. The results from running small-scale and medium-scale problems in MOHVDO and MOSA were compared with the solutions obtained from the epsilon-constraint method. In particular, the error percentage of MOHVDO's objective functions was less than 2% compared to the epsilon-constraint method for all solved problems.



(d): Spacing



(e): NOS



(f): MD

Fig. 4. (*continued*).

For future research, one can develop the problem within a fuzzy environment. An interesting extension would be to develop the problem within a dynamic environment through the design of a real-time order acceptance and analytical scheduling framework able to predict the dynamic production status of flow shops and the corresponding makespan.

CRediT authorship contribution statement

Madjid Tavana: Conceptualization, Methodology, Formal analysis, Writing – review & editing, Validation. **Vahid Hajipour:** Conceptualization, Investigation, Formal analysis, Editing, Visualizations, Validation. **Mohammad Alaghebandha:** Methodology, Formal analysis,

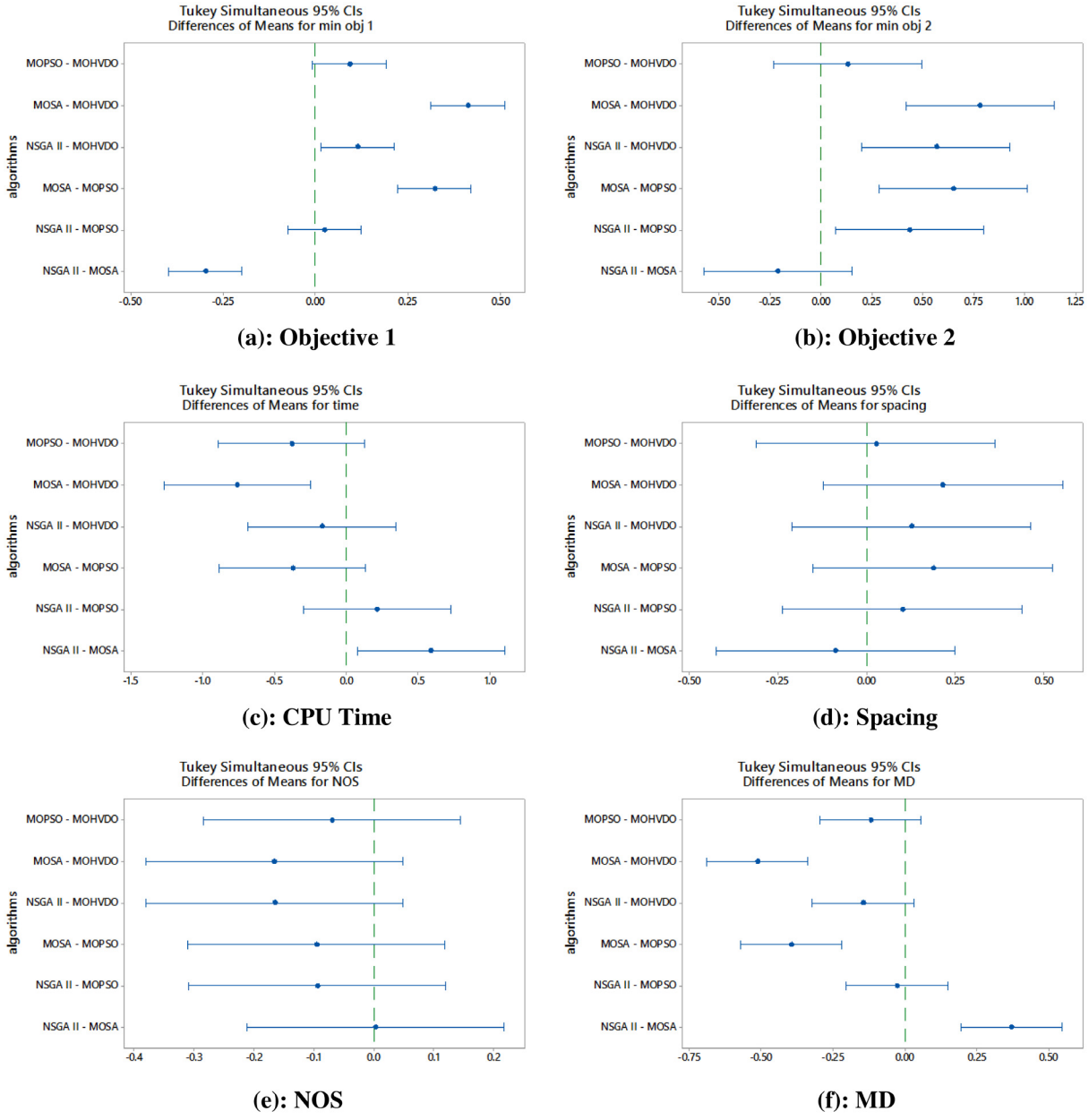


Fig. 5. Interval plot of performance indices.

Writing – review & editing. **Debora Di Caprio:** Methodology, Formal analysis, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

Afzalirad, M., & Rezaeian, J. (2016). A realistic variant of bi-objective unrelated parallel machine scheduling problem: NSGA-II and MOACO approaches. *Applied Soft Computing*, *50*, Article 109123.

Ahmed, F., & Deb, K. (2013). Multiobjective optimal path planning using elitist nondominated sorting genetic algorithms. *Soft Computing*, *17*(7), 1283–1299.

Alaghebandha, M., & Hajipour, V. (2013). A soft computing-based approach to optimize queuing inventory control problem. *International Journal of Systems Science*, *46*(6), 1113–1130.

Amin-Tahmasbi, H., & Tavakkoli-Moghaddam, R. (2011). Solving a bi-objective flow-shop scheduling problem by a multiobjective immune system and comparing with SPEA2+ and SPGA. *Advances in Engineering Software*, *42*(10), 772–779.

An, Y., Chen, X., Gao, K., Zhang, L., Li, Y., & Zhao, Z. (2022). A hybrid multi-objective evolutionary algorithm for solving an adaptive flexible job-shop rescheduling problem with real-time order acceptance and condition-based preventive maintenance. *Expert Systems with Applications*, Article 118711.

Bautista, J., Cano, A., Companys, R., & Ribas, I. (2012). Solving the $F_m|$ block$|$ C_{max} problem using bounded dynamic programming. *Engineering Applications of Artificial Intelligence*, *25*, 1235–1245.

Brammer, J., Lutz, B., & Neumann, D. (2022). Permutation flow shop scheduling with multiple lines and demand plans using reinforcement learning. *European Journal of Operational Research*, *299*(1), 75–86.

Chen, W., Gong, X., Rahman, H. F., Liu, H., & Qi, E. (2021). Real-time order acceptance and scheduling for data-enabled permutation flow shops: Bilevel interactive optimization with nonlinear integer programming. *Omega*, *105*, Article 102499.

Choi, S.-W., & Kim, Y. D. (2009). Minimizing total tardiness on a two-machine re-entrant flowshop. *European Journal of Operational Research*, *199*, 375–384.

Coello, C. A. C., Pulido, G. T., & Lechuga, M. S. (2004). Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, *8*(3), 256–279.

Coello Coello, C. A., Van Veldhuizen, D. A., & Lamont, G. B. (2002). *Evolutionary algorithms for solving multi objective problems*. Kluwer Academic Publishers.

Collette, Y., & Siarry, P. (2003). *Decision engineering series*, *Multiobjective optimization: principles and case studies*. Berlin: Springer.

Companys, R., & Mateo, M. (2007). Different behavior of a double branch-and-bound algorithm on $F_m|$ prmu $|$ c_{max} and $F_m|$ block $|$ C_{max} problems. *Computers & Operations Research*, *34*(4), 938–953.

Czyzak, P., & Jaszkiewicz, A. (1998). Pareto simulated annealing – a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, *6*(7), 34–47.

Doush, I. A., Al-Betar, M. A., Awadallah, M. A., Alyasseri, Z. A. A., Makhadmeh, S. N., & El-Abd, M. (2022). Island neighboring heuristics harmony search algorithm for flow shop scheduling with blocking. *Swarm and Evolutionary Computation*, *74*, Article 101127.

Ferreira, C., Figueira, G., & Amorim, P. (2022). Effective and interpretable dispatching rules for dynamic job shops via guided empirical learning. *Omega*, *111*, Article 102643.

Gong, D., Han, Y., & Sun, J. (2018). A novel hybrid multi-objective artificial bee colony algorithm for blocking lot-streaming flow shop scheduling problems. *Knowledge-Based Systems*, *148*, 115–130.

Grabowski, J., & Pempera, J. (2007). The permutation flow shop problem with blocking, a Tabu search approach. *Omega*, *35*, 302–311.

Hajipour, V., Mehdizadeh, E., & Tavakkoli-Moghaddam, R. (2014). A novel Pareto-based multi-objective vibration damping optimization algorithm to solve multi-objective optimization problems. *Scientia Iranica: Transaction E*, *21*(6), 2368–2378.

Hamzadayı, A. (2020). An effective benders decomposition algorithm for solving the distributed permutation flowshop scheduling problem. *Computers & Operations Research*, *123*, Article 105006.

Han, Y., Gong, D., Jin, Y., & Pan, Q. (2016). Evolutionary multi-objective blocking lot-streaming flow shop scheduling with interval processing time. *Applied Soft Computing*, *42*, 229–245.

Han, Y., Gong, D., Li, J., & Zhang, Y. (2016). Solving the blocking flow shop scheduling problem with makespan using a modified fruit fly optimization algorithm. *International Journal of Production Research*, *54*(22), 6782–6797.

Han, Y., Li, J., Sang, H., Liu, Y., Gao, K., & Pan, Q. (2020). Discrete evolutionary multi-objective optimization for energy-efficient blocking flow shop scheduling with setup time. *Applied Soft Computing*, *93*, Article 106343.

He, Z., Wang, K., Li, H., Song, H., Lin, Z., Gao, K., & Sadollah, A. (2022). Improved Q-learning algorithm for solving permutation flow shop scheduling problems. *IET Collaborative Intelligent Manufacturing*, *4*(1), 35–44.

Huang, K. L. (2008). *Flow shop scheduling with synchronous and asynchronous transportation times*. The Pennsylvania State University.

Huang, K. L., & Ventura, J. A. (2013). Two-machine flow shop scheduling with synchronous material movement. *International Journal of Planning and Scheduling*, *1*(4), 301–315.

Kampmeyer, M., Knust, S., & Waldherr, S. (2016). Solution algorithms for synchronous flow shop problems with two dominating machines. *Computers & Operations Research*, *74*, 42–52.

Khalili, M., & Tavakoli-Moghadam, R. (2012). A multiobjective electromagnetism algorithm for a bi-objective flowshop scheduling problem. *Journal of Manufacturing Systems*, *31*, 232–239.

Khalkhali, A., Mostafapour, M., Tabatabaie, S. M., & Ansari, B. (2016). Multiobjective crashworthiness optimization of perforated square tubes using modified NSGAII and MOPSO. *Structural and Multidisciplinary Optimization*, *54*(1), 45–61.

Kirkpatrick, S., Gelatt, C. D., Jr., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671–680.

Koulamas, C., & Kyparisis, G. J. (2022). Simultaneous minimisation of mean and variation of waiting times in a two-stage proportionate blocking flow shop. *Journal of the Operational Research Society*, 1–11.

Kurdi, M. (2020). A memetic algorithm with novel semi-constructive evolution operators for permutation flowshop scheduling problem. *Applied Soft Computing*, *94*, Article 106458.

Lang, S., Reggelin, T., Schmidt, J., Müller, M., & Nahhas, A. (2021). NeuroEvolution of augmenting topologies for solving a two-stage hybrid flow shop scheduling problem: A comparison of different solution strategies. *Expert Systems with Applications*, *172*, Article 114666.

Liu, Y., Shen, W., Man, Y., Liu, Z., & Seferlis, P. (2019). Optimal scheduling ratio of recycling waste paper with NSGAII based on deinked-pulp properties prediction. *Computers & Industrial Engineering*, *132*, 74–83.

Luo, S. (2020). Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. *Applied Soft Computing*, *91*, Article 106208.

Mehdizadeh, E., Daei Niaki, S. V., & Rahimi, V. (2016). A vibration damping optimization algorithm for solving a new multiobjective dynamic cell formation problem with workers training. *Computers & Industrial Engineering*, *101*, 35–52.

Mehdizadeh, E., Tavakkoli-Moghaddam, R., & Yazdani, M. (2015). A vibration damping optimization algorithm for a parallel machines scheduling problem with sequence-independent family setup times. *Applied Mathematical Modelling*, *39*(22), 6845–6859.

Milacron, C. (1989). T-line machining center alternatives. *Manufacturing Engineering*, *103*(7), 14–15.

Mitra, D., Romeo, F., & Vincentelli, A. S. (1986). Convergence and finite-time behavior of simulated annealing. *Advances in Applied Probability*, *74*, 7–771.

Miyata, H. S., & Nagano, M. S. (2019). The blocking flow shop scheduling problem: A comprehensive and conceptual review. *Expert Systems with Applications*, *137*, 130–156.

Moslehi, G., & Khorasanian, D. (2013). Optimizing blocking flow shop scheduling problem with total completion time criterion. *Computers & Operations Research*, *40*, 1874–1883.

Moslemi, S., Sabegh, M. H. Z., Mirzazadeh, A., Ozturkoglu, Y., & Maass, E. (2017). A multiobjective model for multi-production and multi-echelon closed-loop pharmaceutical supply chain considering quality concepts: NSGAII approach. *International Journal of Systems Assurance Engineering and Management*, *8*(2), 1717–1733.

Nabipoor Afruzi, E., Najafi, A., Roghanian, E., & Mazinani, M. (2014). A multiobjective imperialist competitive algorithm for solving discrete time, cost and quality trade-off problems with mode-identity and resource-constrained situations. *Computers & Operations Research*, *50*, 80–96.

Nam, D., & Park, C. H. (2000). Multiobjective simulated annealing: a comparative study to evolutionary algorithms. *International Journal of Fuzzy Systems*, *2*(2), 87–97.

Narendra, N., & Liang, L. (2008). Identification and systems optimization. *IEEE Transportation Systems*, *125*, 312–325.

Nazif, H. (2021). A new approach for solving the flow-shop scheduling problem using a parallel optimization algorithm. *Journal of Ambient Intelligence and Humanized Computing*, *12*(12), 10723–10732.

Nobari, A., Kheirkhah, A. S., & Hajipour, V. (2018). A Pareto-based approach to optimize reliable aggregate production planning problem. *International Journal of Services and Operations Management*, *29*(1), 59–84.

Pasandideh, S. H. R., & Niaki, S. T. A. (2012). Genetic algorithm in a facility location problem with random demand within queuing framework. *Journal of Intelligent Manufacturing*, *23*, 651–659.

Pasandideh, S. H. R., Niaki, S. T. A., & Hajipour, V. (2013). A multi-objective facility location model with batch arrivals: Two parametric-tuned meta-heuristic algorithms. *Journal of Intelligent Manufacturing*, *24*, 331–348.

Pasandideh, S. H. R., Niaki, S. T. A., & Roozbeh Nia, A. (2011). A genetic algorithm for vendor managed inventory control system of multi-product multi-constraint economic order quantity model. *Expert Systems with Applications*, *38*(3), 2708–2716.

Perez-Gonzalez, P., Fernandez-Viagas, V., & Framinan, J. M. (2020). Permutation flowshop scheduling with periodic maintenance and makespan objective. *Computers & Industrial Engineering*, *143*, Article 106369.

Reddy, M., Ratnam, C., Rajyalakshmi, G., & Manupati, V. K. (2017). An effective hybrid multi objective evolutionary algorithm for solving real time event in flexible job shop scheduling problem. *Measurement*, *114*, 78–90.

Ribas, I., & Companys, R. (2015). Efficient heuristic algorithms for the blocking flow shop scheduling problem with total flow time minimization. *Computers & Industrial Engineering*, *87*, 30–39.

Ribas, I., Companys, R., & Tort-Martorell, X. (2021). An iterated greedy algorithm for the parallel blocking flow shop scheduling problem and sequence-dependent setup times. *Expert Systems with Applications*, *184*, Article 115535.

Safari, G., Hafezalkotob, A., Malekpour, H., & Khalilzadeh, M. (2022). Competitive scheduling in a hybrid flow shop problem using multi-leader–multi-follower game - a case study from Iran. *Expert Systems with Applications*, *195*, Article 116584.

Schaller, J., & Valente, J. (2019). Branch-and-bound algorithms for minimizing total earliness and tardiness in a two-machine permutation flow shop with unforced idle allowed. *Computers & Operations Research*, *109*, 1–11. http://dx.doi.org/10.1016/j.cor.2019.04.017.

Shao, Z., Pi, D., & Shao, W. (2017). Self-adaptive discrete invasive weed optimization for the blocking flow-shop scheduling problem to minimize total tardiness. *Computers & Industrial Engineering*, *111*, 331–351.

Soylu, B., Kirca, O., & Azizoglu, M. (2007). Flow shop-sequencing problem with synchronous transfers and makespan minimization. *International Journal of Production Research*, *45*(15), 3311–3331.

Srinivas, N., & Deb, K. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, *2*(3), 221–248.

Suppapitnarm, A., Seffen, K. A., Parks, G. T., & Clarkson, P. J. (2000). A simulated annealing algorithm for multiobjective optimization. *Engineering Optimization*, *33*(1), 59–85.

Tsai, J., Yang, C. I., & Chou, J. H. (2014). Hybrid sliding level Taguchi-based particle swarm optimization for flow shop scheduling problems. *Applied Soft Computing*, *15*, 177–192.

Ulungu, E. L., Teghem, J. F. P. H., Fortemps, P. H., & Tuyttens, D. (1999). MOSA method: A tool for solving multiobjective combinatorial optimization problems. *Journal of Multi-Criteria Decision Analysis*, *8*(4), 221.

Waldherr, S., & Knust, S. (2014). Complexity results for flow shop problems with synchronous movement. *European Journal of Operational Research*, *242*(1), 34–44.

Waldherr, S., & Knust, S. (2016). Decomposition algorithms for synchronous flow shop problems with additional resources and setup times. *European Journal of Operational Research*, *259*(3), 847–863.

Waldherr, S., Knust, S., & Briskorn, D. (2017). Synchronous flow shop problems: How much can we gain by leaving machines idle? *Omega*, *72*, 15–24.

Wang, L., Pan, Q., Suganthan, P. N., Wang, W., & Wang, Y. (2010). A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems. *Computers & Operations Research*, *37*, 509–520.

Zeng, R., Basseur, M., & Hao, J. (2013). Solving bi-objective flow shop problem with hybrid path relinking algorithm. *Applied Soft Computing*, *13*, 4118–4132.

Zidi, I., Mesghouni, K., Zidi, K., & Ghedira, K. (2012). A multiobjective simulated annealing for the multi-criteria dial a ride problem. *Engineering Applications of Artificial Intelligence*, *25*, 1121–1131.

Zitzler, E., & Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms – a comparative case study. In *International conference on parallel problem solving from nature* (pp. 292–301). Berlin, Heidelberg: Springer.