Management and Production Engineering Review
Volume 14 • Number 1 • March 2023 • pp. 13–24
DOI: 10.24425/mper.2023.145362

Minimizing the Makespan and Total Tardiness in Hybrid Flow Shop Scheduling with Sequence-Dependent Setup Times

Seyyed Mostafa MOUSAVI[1], Parisa SHAHNAZARI-SHAHREZAEI[2]

[1] *Department of Technical and Engineering, Nowshahr Branch, Islamic Azad University, Mazandaran, Iran*
[2] *Department of Industrial Engineering, Central Tehran Branch, Islamic Azad University, Tehran, Iran*

Received:18 February 2022
Acepted: 09 August 2022

Abstract
The paper considers the production scheduling problem in a hybrid flow shop environment with sequence-dependent setup times and the objectives of minimizing both the makespan and the total tardiness. The multi-objective genetic algorithm is applied to solve this problem, which belongs to the non-deterministic polynomial-time (NP)-hard class. In the structure of the proposed algorithm, the initial population, neighborhood search structures and dispatching rules are studied to achieve more efficient solutions. The performance of the proposed algorithm compared to the efficient algorithm available in literature (known as NSGA-II) is expressed in terms of the data envelopment analysis method. The computational results confirm that the set of efficient solutions of the proposed algorithm is more efficient than the other algorithm.

Keywords
Dispatching rules; Genetic algorithm; Hybrid flow shop; Neighborhood search structure.

Introduction

Scheduling is defined as the systematic process of setting various planned activities in order to determine the start and end dates of each activity to execute the whole work in a systematic and orderly or sequence manner. Therefore, scheduling is an important tool in management, manufacturing and engineering to minimize production time and cost. The structure of a production unit in scheduling problems can be classified as single-stage single-machine problems, multi-stage single-machine problems, single-stage multi-machine problems, and multi-stage multi-machine problems. The production unit with the last characteristic is referred to as the hybrid flow shop (HFS) model.

Ruiz and Vazquez-Rodriguez (2010) described the HFS problem in its "standard" form. In the standard form of the HFS problem, the setup times are negligible or included in the processing times. In this research, we define two types of setup times: the first type is sequence-independent setup times (SIST) and the second is sequence-dependent setup times (SDST). If the setup time depends solely on the job to be processed, it is called sequence-independent. On the other hand, in the sequence-dependent type, it depends on both the job and its previous job.

The objective function is a function of planning variables that is minimized or maximized during optimization. The scheduling problems are evaluated during optimization using the objective functions. Due to highly competitive markets, limited resources and just-in-time concepts, the goal of production managers is usually to optimize multiple objective functions. In this research, one of the scheduling objectives is assumed to be the minimization of the maximum completion time (makespan or $C_{\max}$), which often is a standard scheduling objective to increase internal efficiency with the efficient utilization of resources. Another scheduling objective is assumed to be the minimization of the total tardiness, which often is a standard scheduling objective to increase external efficiency by reducing penalties incurred for late jobs. In order to consider the process-based and customer-based objectives, the minimization of two objective functions, including maximum completion time and total tardiness, is intended simultaneously.

Corresponding author: *Seyyed Mostafa Mousavi – Department of Technical and Engineering, Nowshahr Branch, Islamic Azad University, Karimi, Nowshahr, Iran, phone: +46 51686 695, e-mail: mostafa_mousavi85@yahoo.com*

According to the simultaneous approach in objective functions (Collette & Siarry, 2003), both criteria are considered as primary objectives and the target is to search a set of optimal solutions (with other titles, Pareto front, efficient solutions, non-dominated solutions, etc.). The HFS scheduling problem is a strongly NP-hard problem (Ruiz & Maroto, 2006). To develop a solution method, a meta-heuristic based on genetic algorithm (GA) has been proposed to search a set of efficient solutions for investigated problem.

The remainder of the paper is organized as follows: Section 2 gives review of relevant literature. In Section 3; a bi-objective GA is presented for solving HFS scheduling problem. The parameter setting and computational results are presented in Section 4. Finally, Section 5 consists conclusions and future research.

Literature review

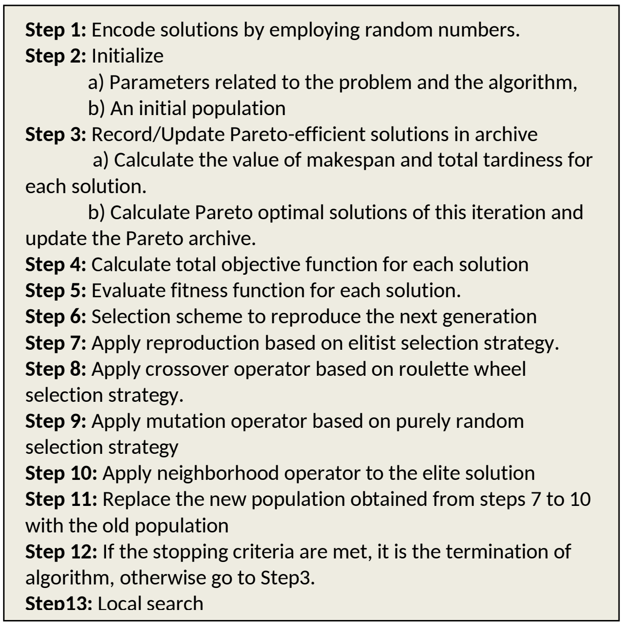
The literature review section refers to several instances of papers on multi-objective HFS scheduling problems solved using GA. Neufeld et al. (In Press) presented a comprehensive review of the literature on multi-objective HFS scheduling.

Dugardin et al. (2010) studied the scheduling problem in the reentrant HFS (RHFS) environment with identical parallel machines. There was a queue before each machine at each stage. These queues were managed by sequence rules. In this paper, the scheduling criteria include (a) minimizing the makespan and (b) maximizing the utilization rate of the bottleneck. They developed a multi-objective GA using the Lorenz dominance relationship to efficiently solve their problem. Abyaneh and Zandieh (2012) focused on the HFS problem with identical parallel machines and two additional attributes including limited buffer spaces and the SDST constraint. The sequence of job was encoded in the first stage, and for the other stages, the jobs were sequenced in the order of their completion time in the previous stage. The first available machine rule was used to assign jobs to machines at each stage. A meta-heuristic approach based on GA is proposed to minimize the makespan and the total tardiness of jobs. Fadaei and Zandieh (2013) considered group scheduling in the HFS scheduling problem with identical parallel machines within the area of sequence-dependent family setup times and two objectives of minimizing makespan and total tardiness of jobs. They focused on three multi-objective algorithms, multi-objective GA, sub-population GA-II and non-dominated sorting GA-II (NSGA-II), to solve the mentioned problem. The sequence of groups and jobs assigned to each group at the first stage were determined based on an encoding method employing random numbers. For other stages, groups and jobs were assigned to machines based on their completion time in the previous stage. Therefore, the first available machine rule was used to assign groups and jobs to machines at each stage. Ebrahimi et al. (2014) investigated the HFS scheduling problem with identical parallel machines, sequence dependent family setup time, and uncertain due dates. The sequence of job was coded in the first stage, and for the other stages, groups and jobs were assigned to the machines based on their completion time in the previous stage. They developed two multi-objective evolutionary algorithms based on GA, namely: NSGA-II and multi-objective GA (MOGA) to minimize the makespan and the total tardiness of jobs. Cho and Jeong (2017) addressed a two-level hierarchical process on production planning and scheduling of the RHFS with identical parallel machines. The lower level scheduling objectives were to minimize the makespan and to minimize the total tardiness. They developed the combination of preemptive goal programming based production planning algorithms and Pareto genetic based scheduling algorithms. Mousavi et al. (2018) examined the scheduling problem in the SDST RHFS environment with identical parallel machines and the learning effect to minimize the total tardiness and the makespan. The job sequence was encoded in the first stage, and for the other stages, the jobs were sequenced in the order of their completion time in the previous stage. The earliest completion time rule was used to assign jobs to machines at each stage. They presented a multi-objective GA somewhat similar to NSGA-II. Li et al. (2019) presented the HFS scheduling problem with identical parallel machines and common due dates. An advanced GA-based on the NSGA-II, was utilized to minimize the total waiting time and the total earliness or tardiness. Chen et al. (2020) studied the scheduling of an energy-efficient HFS with unrelated parallel machines and lot streaming in order to minimize both the production makespan and electric power consumption. They applied NSGA-II to obtain approximate Pareto solutions. Zheng et al. (2021) addressed a flexible flow shop scheduling problem considering limited buffers and step-deteriorating jobs, where there were multiple non-identical parallel machines. To handle this problem, a hybrid meta-heuristic algorithm based on GA, variable neighborhood search and simulated annealing is developed to minimize the makespan and total tardiness simultaneously.

The cost-effectiveness of GA in solving the multi-objective optimization problem (MOP) has led to the emergence of different approaches from the development of GA in the literature. Although much attention has been paid to multi-objective GA, still other approaches of GA development can be proposed to increase the performance of the algorithm. Many of the existing literatures on HFS problems solved using GA have not considered the investigation of all or part of the structure of the proposed solution algorithm in order to increase its effective performance. In this research, the structure of the proposed algorithm has been investigated to achieve a more efficient algorithm. This is the reason for different alternatives have been proposed for each component of the algorithm structure. Then computational tests show the best alternative for each part of the algorithm structure. The initial population, neighborhood search structures and dispatching rules will be investigated in the structure of the proposed algorithm.

The proposed bi-objective genetic algorithm

In this research, the modified GA is presented to solve the bi-objective optimization problem. The flowchart of the proposed bi-objective GA called BOGA and its pseudo code are illustrated in Figures 1 and 2, respectively. In the following, the structure and details of the proposed algorithm are described extensively.



```
Step 1: Encode solutions by employing random numbers.
Step 2: Initialize
        a) Parameters related to the problem and the algorithm,
        b) An initial population
Step 3: Record/Update Pareto-efficient solutions in archive
        a) Calculate the value of makespan and total tardiness for
each solution.
        b) Calculate Pareto optimal solutions of this iteration and
update the Pareto archive.
Step 4: Calculate total objective function for each solution
Step 5: Evaluate fitness function for each solution.
Step 6: Selection scheme to reproduce the next generation
Step 7: Apply reproduction based on elitist selection strategy.
Step 8: Apply crossover operator based on roulette wheel
selection strategy.
Step 9: Apply mutation operator based on purely random
selection strategy
Step 10: Apply neighborhood operator to the elite solution
Step 11: Replace the new population obtained from steps 7 to 10
with the old population
Step 12: If the stopping criteria are met, it is the termination of
algorithm, otherwise go to Step3.
Step13: Local search
```

Fig. 2. Pseudo-code of BOGA

Step 1: Encoding

Encoding is a step to convert a scheduling solution into a chromosome. For our problem, a position in a chromosome (i.e., gene) represents a job number (i.e.,

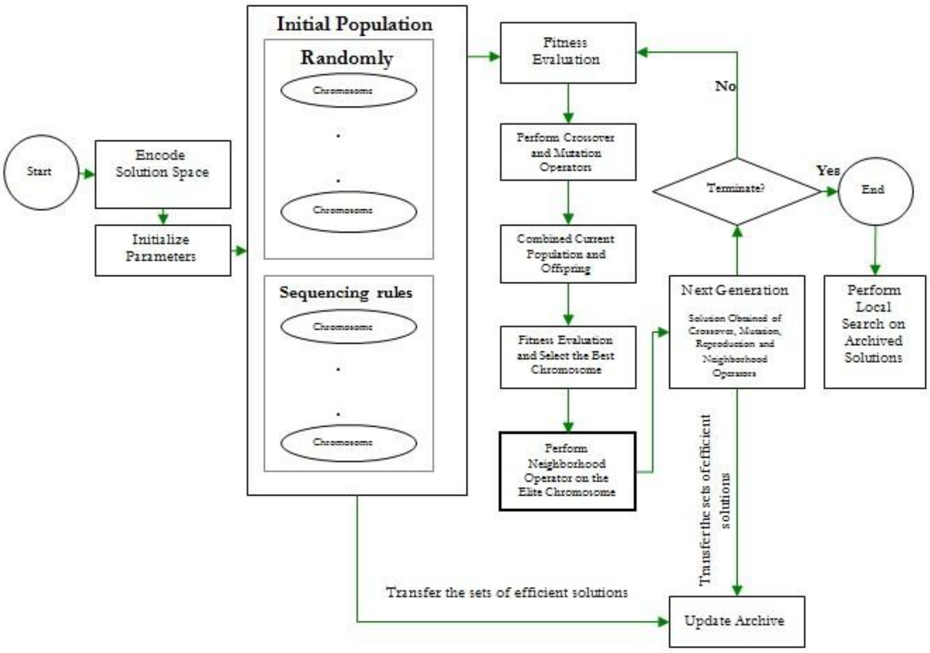

Start
Encode Solution Space
Initialize Parameters
Initial Population
Randomly
Chromosome
.
.
Chromosome
Sequencing rules
Chromosome
.
.
Chromosome
Fitness Evaluation
Perform Crossover and Mutation Operators
Combined Current Population and Offspring
Fitness Evaluation and Select the Best Chromosome
Perform Neighborhood Operator on the Elite Chromosome
No
Terminate?
Yes
End
Next Generation
Solution Obtained of Crossover, Mutation, Reproduction and Neighborhood Operators
Perform Local Search on Archived Solutions
Transfer the sets of efficient solutions
Transfer the sets of efficient solutions
Update Archive

Fig. 1. Flowchart of proposed algorithm

job based representation) and the number of positions (i.e., chromosome size) corresponds to the number of jobs. For example consider a problem with five jobs, two stages, two machines at stage one, three machines at stage two, and with the chromosome as [3 1 4 2 5]. It is known that the machines in parallel are identical in capability and processing rate. Therefore, the first job (3) is scheduled on the machine 1 and second job (1) is assigned to the machine 2 at stage one. Then, the job 4 is assigned to the machine with the earliest completion time. This process continues like this, until all jobs are assigned to the first stage machines. In order to determine the sequence of jobs in the second stage, dispatching rules are used. In this paper, dispatching rules such as first-in-first-out (FIFO) rule, arrival time-remaining processing time (AT-RPT) rule and slack time (SL) rule are presented as a candidate. Consequently one of these rules should be chosen and sequence of jobs is determined based on selected rule. Therefore, one of the research discussions in the computational results section is which of the dispatching rules is more suitable for the sequence of jobs from stage 2 to the last stage. The AT-RPT rule for each job at each stage is calculated as the current time of each job (CT) minus the arrival time of each job (AT) plus the remaining processing time of each job (RPT), where the arrival time of all jobs is assumed to be zero (AT-RPT=CT-AT+RPT). The SL rule for each job at each stage is calculated as the due date of each job (DD) minus the current time of each job minus the remaining processing time of each job (SL=DD-CT-RPT). The priority of the AT-RPT is the largest first and the priority of the SL rule is the smallest first. In determining remaining process time, in addition to processing time, there is setup time which related to the sequence of jobs. Since the jobs sequence of each stage (stage 2 to end) is not clear, the formula obtained by Kurz and Askin (2003) is used to find the approximation of the remaining process time, as follows:

$$\tilde{p}_i^t = p_i^t + \min_{j\in\{0,1,\ldots,n\},j\neq i}(s_{ji}^t) \qquad t = 2,\ldots,g; \quad i = 1,2,\ldots,n \tag{1}$$

$$RPT_i^k = \sum_{t=k}^{g}\left(p_i^t + \min_{j\in\{0,1,\ldots,n\},\ j\neq i}(s_{ji}^t)\right) \qquad k = 2,3,\ldots,g;\ i = 1,2,\ldots,n \tag{2}$$

where

n number of jobs

g number of serial stages

d_i due date for job i

S_{ji}^t dependent setup time from job j to job i at stage t if job j is immediately before job i

S_{0j}^t independent setup time for job j at stage t if job j is first job for scheduling

P_i^t processing time for job i at stage t

$\tilde{P}_i^t$ modified processing time for job i in stage t (this time represents the minimum time at a stage t that must elapse before job i could be completed)

RPT_i^k the sum of modified processing times for job i of stage k to stage g $(k = 2,\ldots,g)$

Step 2: Initialization

i) Initialize the algorithm parameters such as the number of initial population (N_{pop}), Probability of crossover (P_c), Probability of mutation (P_m), Probability of reproduction (P_r), Probability of neighborhood (P_n), Number of generation (N_g). The parameters of proposed BOGA must be set. Therefore, one of the research discussions in the computational results section is to set the parameters of the GA with the Taguchi method. Set the archive with the empty set.

ii) Initialize an initial population (P_0). In this paper, two suggestions are intended to generate an initial population. In the stochastic method, the initial population is generated randomly. In the hybrid method, a part of population is generated by constructive algorithms and remaining part of population is randomly generated. In this paper, constructive algorithms such as earliest due date (EDD), latest due date (LDD), shortest processing time (SPT) and longest processing time (LPT) are presented as a constructive set that produces 2g+4 chromosomes from the initial population. Consequently one of these methods should be chosen and the initial population is generated based on selected method. Therefore, one of the research discussions in the computational results section is which of the two hybrid and stochastic methods is more suitable for the sequence of jobs in stage 1 (or the initial population).

Step 3: Record or Update Pareto-efficient Solutions in Archive

The objective values of all chromosomes in the current population (P_i) are evaluated, then the population of solutions is classified into successive non-dominated fronts. In this method, rank 1 is given to the non-dominated solutions in the population and they are removed from the population; then next set of non-dominated solutions is searched and rank 2 is assigned to them. The process continues until the entire population is ranked. Finally, record the first rank as the Pareto-efficient solutions in this iteration, then the net non-dominated solutions are generated by a set of all non-dominated solutions (in the archive and the first rank). The old Pareto-efficient solutions in the archive are updated with new ones (net non-dominated solutions).

Step 4: Calculate Total Objective Function

The total objective function is constituted as the linear combination of objective functions. Therefore, a weight vector is needed to aggregate the two objectives into a scaling function and also, different weights vectors should be used to define different directions of search. For a solution x, the total objective function in the study is represented as follows:

$$f(x) = \lambda_1 \times f_1(x) + \lambda_2 \times f_2(x) \tag{3}$$

where $f_1(x)$ = makespan; $f_2(x)$ = total tardiness; $\lambda_1 + \lambda_2 = 1$.

The idea behind the λ values is to balance both objectives. For a low value of λ_1, the total tardiness problem will dominate the makespan problem, whereas for a large value of λ_1, the makespan problem will dominate the total tardiness problem.

Step 5: Evaluate Fitness

The original concept of fitness is the larger the better because solutions with larger fitness tend to propagate to the next generation. In this paper, minimization of objectives is considered. Hence it contradicts the original idea of fitness. A transformation should be made to reverse the minimization to maximization. The fitness value of a chromosome, called fitness (x), is given by Eq. 4:

$$\text{fitness}(x) = \frac{1}{f(x)}, \quad x = 1, 2, \ldots N_{\text{pop}} \tag{4}$$

Step 6: Selection Scheme

Roulette wheel, elitist and purely random selections are employed to reproduce the next generation. For the chromosome x with fitness(x), its selection probability, called prob(x), is calculated as follows:

$$\text{prob}(x) = \frac{\text{fitness}(x)}{\sum\limits_{x=1}^{N_{\text{pop}}} \text{fitness}(x)}, \quad x = 1,\ 2, \ldots N_{\text{pop}} \tag{5}$$

Step 7: Reproduction

Based on elitist selection, $N_{\text{pop}} \times P_r$ solutions from current population are selected. It means that $N_{\text{pop}} \times P_r$ solutions with the highest probability in the current population are copied to the next generation.

Step 8: Crossover Operator

Based on roulette wheel selection, $N_{\text{pop}} \times P_c$ pairs of parents from current population are selected, and performed crossover on the parents. One-Point Crossover (1PX) is applied in this paper. To generate an offspring from this operator, a random number is generated and each parent is divided into two parts according to the cut point. The offspring chromosome is obtained from the combination of the first part of parent 1, and the second part of parent 2. If there is a need to modify the chromosome, it will definitely be done. The obtained solutions are transferred to the next generation.

Step 9: Mutation Operator

Based on purely random selection, $N_{\text{pop}} \times P_m$ chromosomes from current population are selected, and mutated individual bits. Swap move is applied in this paper. To generate a sequence from this operator, choose randomly two positions for its insertion (i.e., i and k), insert the job found in the ith position of sequence S in position k of sequence S, insert the job found in the kth position of sequence S in position i of sequence S, and adjust remaining jobs in the sequence accordingly by not changing the relative positions of the other jobs. The obtained solutions are transferred to the next generation.

Step 10: Neighborhood Operator

Neighborhood search structure algorithms move from solution to solution in the search space. The main aim of neighborhood search structure is to produce a new solution from current solution by making a slight change in it.

A neighborhood relation on the search space is defined to generate $N_{\text{pop}} \times P_n$ solutions of an elite solution. It is known that solution with the highest probability is the best solution in the population. In order to select the elite solution, the current population and the solutions achieved from crossover and mutation operators (steps 8 and 9) are combined. Then, solution corresponding to the maximum probability (Eq. (6)), called new_prob(x), is selected as the elite solution. In this paper, several neighborhood relations as inversion move, shift move, swap move, and neighborhood swapping are presented as a candidate. Consequently one of these relations should be chosen and new solutions are produced based on selected structure. Therefore, one of the research discussions in the computational results section is which of the four neighborhood relations is more suitable for this operator. Then, the solutions obtained from this operator are transferred to the next generation.

$$\text{new\_prob}(x) = \frac{\text{fitness}(x)}{\sum\limits_{x=1}^{N_{\text{pop}}(1+P_c+P_m)} \text{fitness}(x)} \qquad (6)$$
$$x = 1, 2, \ldots, N_{\text{pop}}(1 + P_c + P_m)$$

The candidate neighborhood relations are introduced as follows. A candidate solution is presented by its configuration vector $X = (x_1 \ldots x_n)$ and denote by π_i a subsequence of X of arbitrary length, by (x_i) the subsequence consisting of a single configuration x_i, and by "." the concatenation operator. In swap, shift and inversion moves, first chooses randomly two positions (i.e., i and j) in a solution.

$$\begin{aligned}&\text{Swap move}\,(X,i,j)\\ &\quad = \text{SWP}\,(\pi_1 \cdot (x_i) \cdot \pi_2 \cdot (x_j) \cdot \pi_3, i, j)\\ &\quad = \pi_1 \cdot (x_j) \cdot \pi_2 \cdot (x_i) \cdot \pi_3 \end{aligned} \qquad (7)$$

$$\begin{aligned}&\text{Shift move}\,(X,i,j)\\ &\quad = \text{BSH}\,(\pi_1 \cdot (x_i) \cdot \pi_2 \cdot (x_j) \cdot \pi_3, i, j)\\ &\quad = \pi_1 \cdot (x_j).(x_i) \cdot \pi_2 \cdot \pi_3, \quad i < j\\ &\text{Shift move}\,(X,i,j)\\ &\quad = \text{FSH}\,(\pi_1 \cdot (x_i) \cdot \pi_2 \cdot (x_j) \cdot \pi_3, i, j)\\ &\quad = \pi_1 \cdot \pi_2 \cdot (x_j) \cdot (x_i) \cdot \pi_3, \quad i > j \end{aligned} \qquad (8)$$

$$\begin{aligned}&\text{Inversion move}\,(X,i,j)\\ &\quad = \text{Inv}\,(\pi_1 \cdot (x_i)(x_{i+1}) \ldots (x_{j-1})(x_j) \cdot \pi_2, i, j)\\ &\quad = \pi_1 \cdot (x_j)(x_{j-1}) \ldots (x_{i+1})(x_i)\pi_2 \end{aligned} \qquad (9)$$

$$\begin{aligned}&\text{Neighborhood swapping}\,(X)\\ &\quad = \{X' : \; X' = \text{Swap move}\,(X,i,j),\\ &\quad \text{for all} \;\; i = 1, \ldots, n-1, \;\; j = i+1, \ldots, n\} \end{aligned} \qquad (10)$$

Step 11: Replacement

Solutions obtained from the previous steps (include steps 7 to 10) are combined as new population (P_{i+1}).

Step 12: Stopping Rule

If the number of generations equals to the prespecified number (N_g) then stop, otherwise go to step 3.

Step 13: Local Search

The archived solutions (Pareto-efficient solutions) are searched by local search namely random insertion perturbation scheme (RIPS). To explain RIPS considers seed sequence (S) given by {2-3-1-5-4}. The job in the first (i.e., an extreme position) can be inserted at any position to its right. Hence the job in the first position is inserted at any position between 2 and n (here n =5), and a random number generated between 2 and n is used to select the job position. Suppose the selected position is 3. Job 2 is inserted in position 3, yielding a new sequence, S1 as {3-1-2-5-4}. Consider the job in the second (i.e. a non-extreme) position of sequence S and choose randomly two positions for its insertion. Note that this job can be inserted at any position between (2+1)th and nth positions (i.e. a position to its right) and between 1st and (2-1)th positions (i.e. a position to its left). Suppose position 1 is selected to the left and position 4 is selected to the right of job 3. The new sequences thus generated are {3-2-1-5-4} and {2-1-5-3-4}. Call these sequences S2 and S3, respectively. Similarly, for the jobs in positions 3 and 4, we select two positions randomly, one to the right and one to the left, and obtain the resulting sequences {2-1-3-5-4}, {2-3-5-1-4}, {5-2-3-1-4} and {2-3-1-4-5}. Let us call these sequences S4; S5; S6 and S7, respectively. As for the job in the nth position (another extreme position), only one position is randomly selected towards the left of the job, i.e. between positions 1 and n-1. Let the randomly selected position be 2 and the resulting sequence be {2-4-3-1-5}, called S8. After performing the above procedure, for a seed sequence with n jobs, $2 \times (n-1)$ sequences are generated.

The individual feature of this research is to examine the details of the proposed algorithm in order to maximize the algorithm efficiency. In this research due to the increasing amount of computations, only a limited number of alternatives are considered for the main sections of the algorithm, and this is a research limitation.

One of the strengths of the algorithm is to give more important to the fitness function. Because the fitness function determines the best solution in each generation. It is known that solution with the highest probability is the best solution in the population. Also, a new operator called neighborhood operator is defined to search the neighborhoods of best solution in each generation. Then, neighborhood operator is performed to generate a part of solutions of the next generation with search the neighborhoods of best solution in each generation. This means that the new operator is executed only on a specific solution that has been introduced by the fitness function. We hope to get better results by making small changes in the best solution (closest to the obtained solution to the optimal solutions). These slight changes are done through neighborhood operator.

Computational experiments

The required data for the investigated problem consist of number of jobs, number of stages, number of machines, the range of processing times, range of setup times, and range of due date. In order to evaluate the proposed algorithm, the problems created in Mousavi et al. (2011) have been used.

In the MOP literature, many of the performance criteria provided to assess the quality of the non-dominated solutions. In this paper, the introduced method by Ruiz-Torres and Lopez (2004) (namely, FDH approach) is used to compare several efficient sets quantitatively.

In the rest of this section, it includes parameter settings with the Taguchi method, the results of studying the structure of the proposed algorithm, the implementation of the data sets by the proposed algorithm and the algorithm in the literature, and then the results of the comparisons are presented.

Parameter setting

When using algorithm to achieve efficient solutions, the values of algorithm parameters vary depending on the type of problem. Therefore, appropriate value selection has a significant effect on the performance of the algorithm. In this paper, the parameters in the algorithm are determined by the Taguchi method. The Taguchi method implements a quality loss function for evaluating product quality with an orthogonal array to reduce the number of tests.

In setting the parameters, first the control factors, their levels and an orthogonal array for these control factors are selected. The control factors include the inner array. In this paper, the parameters and their levels are shown in Table 1. The square matrix with 4 parameters in 3 levels used in the Taguchi met od is L_9.

Experiments are performed several times for each combination of control factor settings (the inner array). The response data from each run in the outer array is usually placed in the row corresponding to the control factors set in the inner array. Then, the response data in each row of the outer array is transmitted as S/N values. Because the goal of the research is to minimize the makespan and the total tardiness of jobs, S/N ratio must be calculated using lower-is-better formula as Eq. (11).

Table 1
Parameters, levels and S/N ratio

	A (N_{pop})		B (P_c)		C (P_n)		D (N_g)	
	Level	S/N ratio	Level	S/N ratio	Level	S/N ratio	Level	S/N ratio
1	60	3.2318	0.50	2.9810	0.10	3.3415	80	2.6914
2	80	2.9909	0.60	3.0169	0.15	2.5634	90	2.7598
3	100	2.8596	0.70	2.7185	0.20	2.8321	100	2.8071
Delta		0.3722		0.2984		0.5094		0.1157
Rank	2		3		1		4	

$$\frac{S}{N} = -10 \log \left(\frac{\sum_{i=1}^{n} Y_i^2}{n} \right) \tag{11}$$

The problem is that the response data from each run in MOP is usually in the form of sets (efficient solutions). Since the Taguchi function must be evaluated by a criterion, then a function that represents a combination of all objectives is defined as follows (Eq. (12)):

$$Y_i = \frac{\sum_{s=1}^{a_i} \left(\lambda \times C^s_{\max i} + (1-\lambda) \times \overline{T}^s_i \right)}{a_i} \tag{12}$$

where a_i is the number of obtained non-dominated solutions from i-th combinations in L9. $C^s_{\max i}$ and $\overline{T}^s_i$ are, respectively, the makespan and total tardiness values of the solution s in the reference set a_i. λ denotes the weight (or relative importance) given to $C^s_{\max i}$ and $\overline{T}^s_i$.

Also, Table 1 shows the data converted to S/N value. It can also be seen in Table 1, factor C (probability of neighborhood operator) is prominent in the implementation process of determining the parameters of the GA. In addition, the effect of four factors on minimizing makespan and the total tardiness in GA is, in the order of: probability of neighborhood operator, number of initial population, probability crossover operator, and number of generation. The optimal factor combination is A: 100; B: 0.7; C: 0.15; and D: 80.

Studying the initial population in the structure of the proposed algorithm

Since the output results are strongly dependent on the initial set and in order to achieve more efficient solutions, two suggestions (hybrid and stochastic methods) are considered to generate an initial population. First, a part of the population is generated using constructive algorithms such as EDD, LDD, SPT, and LPT (Eq. (13) to (18)) and the rest of the population is randomly generated. This is known as hybrid method. Second, the initial population is generated randomly. This is known as stochastic method.

$$\text{EDD}: \quad d_{[1]} \le d_{[2]} \le \ldots \le d_{[n-1]} \le d_{[n]} \tag{13}$$

$$\text{LDD}: \quad d_{[1]} \ge d_{[2]} \ge \ldots \ge d_{[n-1]} \ge d_{[n]} \tag{14}$$

$$\begin{aligned}\text{SPT}: \quad & p_{[1]}^1 + p_{[1]}^2 + \ldots + p_{[1]}^g \le \ldots \\ & \le p_{[n]}^1 + p_{[n]}^2 + \ldots + p_{[n]}^g\end{aligned} \tag{15}$$

$$\begin{aligned}\text{LPT}: \quad & p_{[1]}^1 + p_{[1]}^2 + \ldots + p_{[1]}^g \ge \ldots \\ & \ge p_{[n]}^1 + p_{[n]}^2 + \ldots + p_{[n]}^g\end{aligned} \tag{16}$$

$$\begin{aligned}\text{SPT}^{\text{t}}: \quad & p_{[1]}^t \le p_{[2]}^t \le \ldots \le p_{[n-1]}^t \le p_{[n]}^t \\ & t = 1, \ldots, g\end{aligned} \tag{17}$$

$$\begin{aligned}\text{LPT}^{\text{t}}: \quad & p_{[1]}^t \ge p_{[2]}^t \ge \ldots \ge p_{[n-1]}^t \ge p_{[n]}^t \\ & t = 1, \ldots, g\end{aligned} \tag{18}$$

The FDH approach is used to evaluate the output results of two mentioned methods above. The degree of efficiency is calculated using equal weights for each of the objectives (0.50). The results are shown in Table 2. Output results indicate that the FDH has the highest score for the hybrid method than the stochastic method. Therefore, the hybrid method is more effective than the other method for generating the initial population. Also, Fig. 3 shows the average efficiency of the hybrid method (case 1) and the stochastic method (case 2) in different levels of the number of jobs and stages. It can also be seen in Fig. 2, the first case has a higher average efficiency than the second case in all situations. This figure illustrates and confirms the conclusion derived from the numerical results based on the performance criterion.

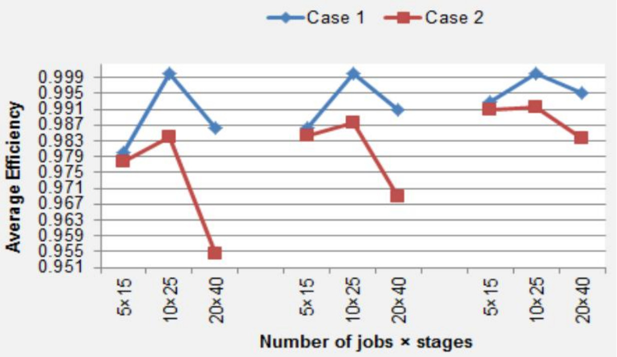


Fig. 3. Average efficiency in the different levels of the number of jobs and stages

Table 2
Result of efficiency of hybrid and stochastic methods

Problem	n	g	τ	R	Hybrid	Stochastic
S1	15	5	0.2	0.2	0.9880	0.9857
S2			0.2	0.8	0.9458	0.9433
S3			0.5	0.5	1.0000	1.0000
S4			0.8	0.2	0.9979	0.9977
S5			0.8	0.8	1.0000	0.9945
M1	25	10	0.2	0.2	1.0000	0.9892
M2			0.2	0.8	1.0000	0.9925
M3			0.5	0.5	1.0000	0.9923
M4			0.8	0.2	1.0000	0.9873
M5			0.8	0.8	1.0000	0.9773
L1	40	20	0.2	0.2	0.9578	0.8653
L2			0.2	0.8	1.0000	0.9917
L3			0.5	0.5	0.9981	0.9977
L4			0.8	0.2	0.9993	0.9978
L5			0.8	0.8	0.9989	0.9920

Studying the neighborhood operator and dispatching rule in the structure of the proposed algorithm

In this subsection, the best alternatives are determined for the dispatching rule and neighborhood operator. The dispatching rules such as FIFO, ATRPT, and SL are selected as a candidate. In order to find the best neighborhood operator, inversion move, shift move, neighborhood swapping and swap move are considered as a candidate. It is known that performance measure (FDH approach) provides better results if the number of algorithms is increased (Ruiz-Torres and Lopez, 2004). This is the reason for combinations of dispatching rules and neighborhood operators have been created. Table 3 shows the twelve combinations obtained with their names and abbreviation codes. The aim of the section is to find the best combination of dispatching rule and neighborhood operator.

The degree of efficiency is calculated using equal weights for each of the objectives. The weight of the objectives in the total objective function, which consists of a linear combination of the objective functions, is considered equal. It is known that the evaluation method is able to rank algorithms through numerical evaluations based on an index namely efficiency. Table 4 represents the results of FDH approach for various problems. In Table 4, the efficiency of GA2 is equal to 1 (the highest efficiency) for S1 problem and

Table 3
Candidate algorithm

No	Abbreviation	Neighborhood operator	Dispatching rule
1	GA1	Inversion move	FIFO
2	GA2	Shift move	FIFO
3	GA3	Neighborhood swapping	FIFO
4	GA4	Swap move	FIFO
5	GA5	Inversion move	AT-RPT
6	GA6	Shift move	AT-RPT
7	GA7	Neighborhood swapping	AT-RPT
8	GA8	Swap move	AT-RPT
9	GA9	Inversion move	SL
10	GA10	Shift move	SL
11	GA11	Neighborhood swapping	SL
12	GA12	Swap move	SL

Table 4
Heuristics efficiency

Problem	Algorithm											
	GA1	GA2	GA3	GA4	GA5	GA6	GA7	GA8	GA9	GA10	GA11	GA12
S1	0.9462	1.0000	0.9966	0.9897	0.9730	0.9706	0.9881	0.9886	0.9178	0.9906	0.9662	0.9580
S2	0.9030	0.9623	0.9757	0.9963	O.8090	0.9739	0.9744	0.9402	0.9844	0.8853	0.9917	0.9974
S3	0.9494	1.0000	0.9931	0.9832	0.9505	0.9753	0.9862	0.9889	0.9821	0.7716	0.9906	0.9923
S4	0.9941	1.0000	0.9969	0.9990	0.9944	0.9967	0.9962	0.9976	0.9906	0.9939	0.9963	0.9961
S5	0.9935	0.9973	0.9966	0.9988	0.9950	1.0000	0.9981	0.9985	0.9970	0.8984	0.9955	0.9972
M1	0.9178	0.9773	0.9437	1.0000	0.9768	0.9733	0.9833	0.9762	0.9963	0.7963	0.9999	0.8615
M2	0.9596	1.0000	0.9888	0.9987	0.9679	0.9824	0.9869	0.9940	0.995O	0.8232	0.9981	0.9955
M3	0.9783	0.9971	0.9925	1.0000	0.9745	0.9945	0.9957	0.9880	0.9934	0.8630	0.9984	0.9311
M4	0.9915	0.9957	0.9940	1.0000	0.9917	0.9924	0.9934	0.9949	0.9970	0.9944	0.9927	0.9811
M5	0.9843	0.9938	0.9910	0.9662	0.9783	0.9768	0.9912	0.9848	0.9981	0.9988	0.9977	0.7090
L1	0.6646	0.9997	0.9980	1.0000	0.7328	0.6367	0.9079	0.7202	0.9888	0.6602	0.9948	0.9976
L2	0.9736	1.0000	0.9783	0.9898	O.9809	0.9940	0.9953	0.9968	0.9949	0.9543	0.9974	0.9324
L3	0.9934	O.9990	O.9990	0.9991	0.9981	0.9935	0.9942	0.9948	0.9961	0.9974	0.995O	0.9421
l4	0.9987	0.9948	0.9976	1.0000	0.9940	0.9979	0.9962	0.9977	0.9967	0.9896	0.9971	0.9985
L5	O.9806	0.9971	0.9981	0.9975	0.9711	0.9893	0.9957	0.9943	0.9986	0.8821	0.9980	0.9989

is better than others. The highest rank (i.e., rank 1) is assigned to GA2 in comparison to other algorithms. Based on the codes in Table 3, GA2 was created from the combination of FIFO and shift move. It means that "FIFO" and "shift move" are classified in rank 1 as the best alternative for the dispatching rule and neighborhood operator respectively. Another example, the efficiency of GA4 is equal to 1 for M4 problem and is better than others. Based on the codes in Table 3, GA4 was created from the combination of FIFO and swap move. It means that "FIFO" and "swap move" are classified in rank 1 as the best alternative for the dispatching rule and neighborhood operator respectively.

Then, several sets of different weights vectors are used to search different directions of space solution (λ_1 = 0.25, 0.5, and 0.75). After running the algorithms for different weights and calculating the efficiency of the algorithms by FDH approach, the results are summarized in Table 5. Table 5 shows that several times (but as percentage) each alternative (neighborhood operator and dispatching rule) is classified in the rank 1. It can also be seen in Table 5 (section I), the FIFO rule with the highest percentage is suggested to find the jobs sequence of the second to end stages. It can also be seen in Table 5 (section II), the shift move with the highest percentage is suggested to generate a section of solutions of the next generation.

Table 5
The summarized results (%)

$\lambda_1 = 0.75$	$\lambda_1 = 0.5$	$\lambda_1 = 0.25$	Alternative	Section
74	74	80	FIFO	I (Dispatching rule)
13	6	6	AT-RPT	
17	20	20	SL	
0	0	0	Inversion move	II (Neighborhood operator)
74	47	57	Shift move	
9	0	23	Neighborhood swapping	
20	53	23	Swap move	

The efficiency of the proposed algorithm

The performance of the proposed BOGA is compared with a NSGA-II algorithm in the literature (Deb et al. 2002). It is noticeable that all of algorithms are implemented in MATLAB 2009a, and run on a PC with 2.30 GHz Intel Core and 4 GB of RAM memory. To show the efficiency and effectiveness of the proposed algorithm in comparison with a NSGA-II, computational experiments were done on various test problems (i.e. small, medium and large). Each test problem has been executed ten times and the "meta-heuristic efficiency" has been calculated with the FDH approach for each algorithm.The comparisons are performed on the basis of the sets of non-dominant solutions obtained by each algorithm.

The results of heuristic efficiency based on the FDH approach for each algorithm are shown in Table 6. Table 6 shows that FDH assigned the highest score to proposed algorithm in comparison with the NSGA-II algorithm in most problems. Also, Fig. 4 depicts the average efficiency of algorithms in different levels of number of jobs and stages. It can be seen that the BOGA has the higher average efficiency than algorithm in the literature for all sizes of the problem. Therefore, the performance of the proposed method is independent of the size of the problem instances.

Table 6
Heuristics efficiency

Problem	n	g	τ	R	BOGA	NSGA-II
S1	15	5	0.2	0.2	0.9991	0.9795
S2			0.2	0.8	0.9715	0.9344
S3			0.5	0.5	1.0000	0.9653
S4			0.8	0.2	0.9995	0.9982
S5			0.8	0.8	0.9945	0.9951
M1	25	10	0.2	0.2	0.9979	0.9968
M2			0.2	0.8	0.9953	0.9939
M3			0.5	0.5	0.9977	0.9967
M4			0.8	0.2	0.9989	0.9998
M5			0.8	0.8	0.9790	0.9782
L1	40	20	0.2	0.2	0.9508	0.9091
L2			0.2	0.8	0.9980	0.9960
L3			0.5	0.5	0.9983	0.9983
L4			0.8	0.2	0.9970	0.9991
L5			0.8	0.8	0.9985	0.9772

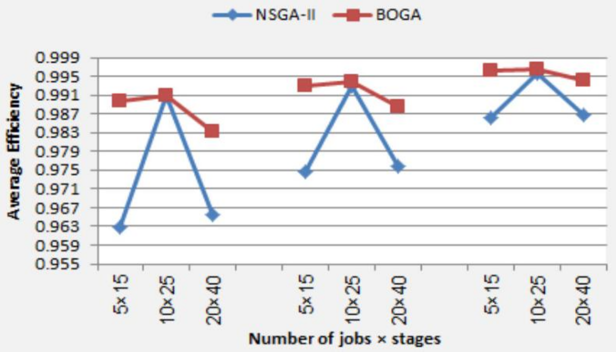


Fig. 4. Average efficiency of algorithms

Now, computational results are expressed in terms of qualitative metrics. Number of Pareto solutions (NPS) criterion presents the number of non-dominated solutions obtained from each algorithm. The larger the number, the better the performance of the algorithm will be. Also, the net non-dominated solutions (NDS) are generated by a set of all non-dominated solutions obtained from all algorithms (whose members should be also non-dominated in relation to one another). The results of NPS and NDS are shown in the Table 7. The values of NPS of BOGA and NSGA-II in the first row of Table 7 are equal to 8 and 6 respectively. As a result, the BOGA front has more non-dominated solutions than NSGA-II front, corresponding to the value of NPS. The results shown in NPS column only evaluate the number of non-dominated solutions found by each algorithm, not their quality. However, the quality of solutions can be measured by NDS. The values of NDS of BOGA and NSGA-II in the first row of Table 7 are equal to 7 and 3 respectively. The larger value of NDS, the better of solution quality we have. As shown in Table 7, the proposed algorithm is more effective than the NSGA-II algorithm in terms NDS and NPS for small, medium and large-sized problems.

Table 7
Number of efficient scheduling

Problem	NPS		NDS	
	BOGA	NSGA-II	BOGA	NSGA-II
S1	8	6	7	3
S2	19	15	10	10
S3	5	7	5	2
S4	10	4	8	3
S5	6	7	4	1
M1	10	6	9	4
M2	10	6	8	3
M3	9	16	8	9
M4	8	5	4	4
M5	8	9	7	3
L1	8	7	7	2
L2	6	4	5	2
L3	8	9	4	4
L4	8	8	2	6
L5	9	6	8	1

Graphical representation is provided to demonstrate output results of the BOGA and NSGA-II. Figure 5 represents the non-dominated solutions of a single run by proposed algorithm and NSGA-II for S2 problem. It is obvious that a "net non-dominated front" is obtained the non-dominated sets of both algorithms. The results indicate that the solutions obtained of BOGA have relatively better quality.

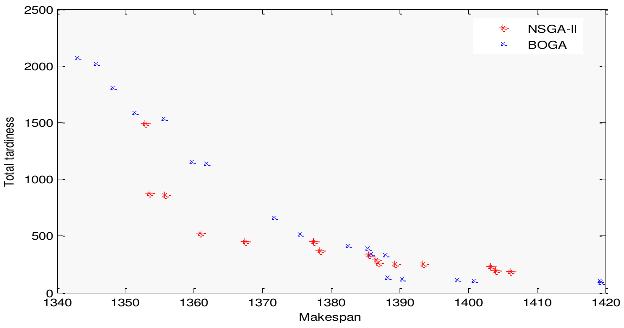


Fig. 5. The non-dominated solutions of BOGA and NSGA-II for S2 problem

Conclusions and future work

This paper considers the problem of scheduling jobs in a hybrid flow shop with the objectives of minimizing both the makespan and the total tardiness. A bi-objective genetic algorithm is proposed for solving this bi-objective optimization problem. Taguchi method is applied to set the parameters of the proposed algorithm. The results show that the neighborhood operator is known as a prominent factor in the process of setting parameters of the BOGA. A new operator named "neighborhood operator" was placed as a new idea in the structure of genetic algorithm. In this paper, two suggested methods for generating the initial population was also studied to improve the quality of the non-dominant front. The results showed that the hybrid method is more effective for generating initial population. Other results obtained from the study of the structure of the proposed algorithm are as follows. The FIFO rule gives better results than the other dispatching rules for jobs sequence from the second to the next stage. The shift move algorithm is the best option among neighborhood structures studied for the neighborhood operator. Finally, the performance of the proposed algorithm is compared with the efficient algorithm in literature (NSGA-II). The efficiency result has been observed that the proposed algorithm provides more efficient solutions than the NSGA-II algorithm. There are two main reasons for the relatively better performance of the proposed algorithm. First, the structure of the proposed algorithm was examined and then it was compared with another algorithm. Second, a neighborhood operator placed in the structure of this algorithm (as a new idea) has had a significant effect on increasing the efficiency of the algorithm. For future research, dispatching rules can be designed based on the objectives and assumptions in the research.

References

Abyaneh S.H. and Zandieh M. (2012), Bi-objective hybrid flow shop scheduling with sequence-dependent setup times and limited buffers. *The International Journal of Advanced Manufacturing Technology*, Vol. 58, No. 1, pp. 309–325.

Chen T.-L., Cheng C.-Y. and Chou Y.-H. (2020), Multi-objective genetic algorithm for energy-efficient hybrid flow shop scheduling with lot streaming. *Annals of Operations Research*, Vol. 290, No. 1, pp. 813–836. DOI: 10.1007/s10479-018-2969-x.

Cho H.-M. and Jeong I.-J. (2017), A two-level method of production planning and scheduling for bi-objective reentrant hybrid flow shops. *Computers & Industrial Engineering*, Vol. 106, No. 3, pp. 174–181.

Collette Y. and Siarry P. (2003), Multiobjective optimization: Principles and case studies. Springer, Berlin Heidelberg New York.

Deb K., Pratap A., Agarwal S. and Meyarivan T. (2002), A fast and elitist multi-objective genetic algorithm: NSGA II. *IEEE Transaction on Evolutionary Computation*, Vol. 6, No. 2, pp. 182–197.

Dugardin F., Yalaoui F. and Amodeo L. (2010), New multi-objective method to solve reentrant hybrid flow shop scheduling problem. *European Journal of Operational Research*, Vol. 203, No. 1, pp. 22–31.

Ebrahimi M., Fatemi Ghomi S. and Karimi B. (2014), Hybrid flow shop scheduling with sequence dependent family setup time and uncertain due dates. *Applied Mathematical Modelling*, Vol. 38, No. 1, pp. 2490–2504.

Fadaei M. and Zandieh M., (2013), Scheduling a bi-objective hybrid flow shop with sequence-dependent family setup times using metaheuristics. *Arab Journal Science Engineering*, Vol. 38, No. 8, pp. 2233–2244.

Kurz M.E. and Askin R.G., (2003), Scheduling flexible flow lines with sequence-dependent setup times. *European Journal of Operational Research*, Vol. 159, No. 1, pp. 66–82.

Li Z., Zhong R.Y., Barenji A.V., Liu J.J., Yu C.X. and Huang G.Q. (2019), Bi-objec-tive hybrid flow shop scheduling with common due date. *Operational Research*, Vol. 181, No. 9, pp. 1–26.

Mousavi S.M., Zandieh M. and Amiri M., (2011), An efficient bi-objective heuristic for scheduling of hybrid flow shops. *International Journal of Advanced Manufacturing Technology*, Vol. 54, No. 1, pp. 287-307.

Mousavi S.M., Mahdavi I., Rezaeian J. and Zandieh M. (2018), An efficient bi-ob-jective algorithm to solve re-entrant hybrid flow shop scheduling with learning effect and setup times. *Operational Research*, Vol. 18, No. 1, pp. 123–158.

Neufeld J.S., Schulz S. and Buscher U., (In Press), A systematic review of multi-objective hybrid flow shop scheduling. *European Journal of Operational Research*. DOI: 10.1016/j.ejor.2022.08.009.

Ruiz R. and Marato C., (2006), A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *European Journal of Operational Research*, Vol. 169, No. 3, pp. 781–800.

Ruiz R. and Vázquez-Rodríguez J.A., (2010), The Hybrid Flow Shop Scheduling Problem. *European Journal of Operational Research*, Vol. 205, No. 1, pp. 1–18.

Ruiz-Torres A.J. and Lopez F.J., (2004), Using the FDH formulation of DEA to evaluate a multi-criteria problem in parallel machine scheduling. *Computers and Industrial Engineering*, Vol. 47, No. 2–3, pp. 107–121.

Zheng Q.-Q., Zhang Y., Tian H.-W., and He L.-J. (2021), An effective hybrid meta-heuristic for flexible flow shop scheduling with limited buffers and step-deteriorating jobs. *Engineering Applications of Artificial Intelligence*, Vol. 106, p. 104503.