International Journal of Production Economics 193 (2017) 832–844

Contents lists available at ScienceDirect

International Journal of Production Economics

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

Makespan minimization in flowshop batch processing problem with different batch compositions on machines

Hossein N.Z. Matin [a], Nasser Salmasi [b,*], Omid Shahvari [c]

[a] Department of Industrial and Enterprise System Engineering, University of Illinois at Urbana-Champaign, Champaign, IL, USA
[b] Corning Incorporated, Wilmington, NC, USA
[c] Department of Mechanical, Industrial, and Manufacturing Engineering, Oregon State University, Corvallis, OR, USA

ARTICLE INFO

Keywords:
Flowshop scheduling
Batch processing
Minimization of makespan
Particle swarm optimization
Mathematical modeling

ABSTRACT

In this research, we consider the flowshop batch processing problem (FBPP) with minimization of makespan, in which the composition of batches can change on different machines. A batch capacity of a machine restricts not only the maximum number of jobs, but also the total attribute size of jobs assigned to the batch processed on the machine. This is the first time that the FBPP is considered for different batch compositions on machines with respect to both the total size and the number of jobs assigned to batches. We propose a mixed-integer linear programming model for the research problem. Since this problem is shown to be NP-hard, several meta-heuristic algorithms based on particle swarm optimization (PSO), enhanced with local search structures, are proposed to solve the research problem heuristically. To have more diversity, different rules are implemented to generate the initial population of the PSO algorithms. Two lower bounding mechanisms are also proposed to generate good quality lower bounds for special cases of the research problem and, consequently, evaluate the performance of the proposed PSO algorithms. A data generation mechanism has been developed in a way that it fairly reflects the real industry requirements. The proposed PSO algorithms are examined by different numerical experiments and the results affirm the efficiency of the proposed algorithms.

1. Introduction

Batch processing problem (BPP) can be observed in many industries and service sectors. In the BPP, the machine processes a batch of jobs at the same time according to its capacity. This capacity is defined based on the restriction on the machine with respect to the number of jobs in a batch and/or the total size of '*jobs' attributes*' like weight or volume. In the BPP, when the process of a batch starts on a machine, the jobs assigned to the batch might not be available as long as the process of the batch on the machine is completed. The number of batches required to process all jobs as well as the number and the type of jobs assigned to each of these batches on a particular machine determine the batch composition on that machine. Since all jobs are processed through production line in unidirectional passes, all machines are placed as a flow-line arrangement based upon jobs' processing plans, i.e., flowshop environment.

Damodaran et al. (2007) discuss several industrial applications of BPP such as wafer fabrication, metalworking, environmental stress screening, chemical treatment of rims, and chemical processes in kilns. In addition to manufacturing applications, the BPP has applications in service sectors and healthcare as well. The BPP can be classified in two different ways based on *the batch processing time* and *the batch capacity restriction*. These classifications are explained as follows:

1.1. Classification of batch processing problems based on the batch processing time

The BPP can be classified into three major categories based on the time required to process the jobs belonging to each batch as follows:

Category A: In this category, it is assumed that a fixed duration of time is required to process the jobs assigned to each batch. The details of this problem can be found in Baker and Trietsch (2009). As an example of application of such problem, consider the process of preparing operating rooms' tools in a hospital. The tools used for the surgery during daily operations should be heated in a furnace for a specific period of time to prepare them for the next day operations.

Category B: In this category, the processing time of each batch on a resource is a function of jobs' attribute such as the sum of the processing time of the jobs assigned to the batch. This problem is called a serial

* Corresponding author.
E-mail addresses: nickzin2@illinois.edu (H.N.Z. Matin), salmasin@corning.com (N. Salmasi), shahvaro@oregonstate.edu (O. Shahvari).

https://doi.org/10.1016/j.ijpe.2017.09.015
Received 2 April 2017; Received in revised form 6 August 2017; Accepted 22 September 2017
Available online 23 September 2017

batch scheduling problem where the jobs assigned to batches are processed consecutively. The application of such problem is in the steel-making process in steel plant. Tang and Liu (2009), Shen et al. (2014), and Shahvari and Logendran (2017, 2016) approach this problem and proposed different mathematical models and solution methods.

Category C: In the third category, the processing time of each batch is equal to the largest processing time of the jobs assigned to the batch. Most of the research performed in the BPP belong to this category since it has many applications in real world industries. Baker and Trietsch (2009) address a burn-in operations problem for electronic components as an example of this category.

1.2. Classification of batch processing problems based on the batch capacity restriction

The BPP are categorized into two major categories based on the existence of any restriction on the number of jobs assigned to a batch.

Category 1: There is no restriction in the batch capacity (both the size capacity and the number of jobs assigned to a batch). These problems are observed when the items are very small. Baker and Trietsch (2009) discuss about interesting properties of these problems.

Category 2: There is a restriction(s) in the batch capacity. The restriction can be based on the maximum number of jobs assigned to a batch or the existence of a size capacity for the batch. Sometimes, both the maximum number of jobs and the size capacity of a batch should be considered at the same time. For instance, consider the trucks used to transfer the final products of a car manufacturing company, as automobiles, toward customers. In this case, each truck has a fixed number of slots for cars. At the same time, each truck has a capacity based on the weight of the loaded cars. Thus, cars assigned to each truck for transportation are chosen based on both restrictions. If the truck is transferring small size cars, then the maximum number of slots in the truck is limiting the number of cars assigned to the truck. But when the truck is transferring heavier cars such as SUVs, then the truck can transfer less cars than its number of available slots.

1.3. Related works

Potts and Kovalyov (2000) provided a literature review for the BPP proposed before 2000. Uzsoy (1994) studied the single machine BPP in which the processing time of each batch is equal to the maximum processing time of the jobs assigned to the batch (category C). He assumed that each batch is restricted by only the size capacity. He introduced several heuristic algorithms with minimization of makespan and minimization of total completion times as the criterion. Damodaran et al. (2007) approached the single machine BPP in which the machine has both types of restrictions on the maximum number of jobs and the size capacity. They assumed that the processing time of each batch is equal to the maximum processing time of the jobs assigned to the batch. They proposed a mixed-integer linear programming (MILP) model to solve the problem. Since the research problem was NP-hard, they proposed a meta-heuristic algorithm based on simulated annealing (SA) to heuristically solve the problem. Xu et al. (2012) addressed the single machine BPP with dynamic arrivals of jobs. They assumed that the capacity of the machine is based on job sizes and the processing time of each batch is equal to the largest processing time of the jobs assigned to that batch. They proposed a meta-heuristic algorithm based on ant colony optimization to solve the research problem heuristically.

Damodaran and Srihari (2004) approached the two-machine flowshop batch processing problem (FBPP) with the size capacity restriction for the first time. The processing time of each batch is the maximum processing time of the jobs assigned to the batch. They proposed two MILP models to minimize makespan with unlimited or zero intermediate buffer. They assumed that the batch composition remains the same on all machines. This assumption is valid in many industries such as chemical and food industries. Liao and Liao (2008) proposed two MILP models in both unlimited and zero intermediate buffer cases for the two-machine FBPP. They showed that their proposed mathematical models are more efficient than Damodaran and Srihari (2004) models. Manjeshwar et al. (2009) and Liao and Huang (2011) proposed two meta-heuristic algorithms based on SA and tabu search (TS), respectively, to heuristically solve the same problem proposed by Damodaran and Srihari (2004). Tang and Liu (2009) studied the two-machine FBPP in which the jobs have release times by considering the minimization of makespan as the criterion. They assumed that the first machine is a discrete and the second one is a serial batch processor. They proposed a MILP model and developed a dynamic programming-based heuristic algorithm to solve the research problem. Sung and Yoon (1997) addressed the two-machine FBPP with minimization of makespan as the criterion in which the jobs have dynamic release times. They considered a fixed processing time for each batch (category A).

Sung et al. (2000) approached a multi-stage FBPP for the first time. They assumed that the processing time of each batch on each machine is fixed (category A) and each machine has a capacity on the number of jobs. They studied several dominant criteria to reduce the complexity of the problem.

A summary of related publications in different shop environments is shown in Table 1 with respect to the batch composition, batch processing time, and batch capacity restrictions.

1.4. Motivation and contribution

In all available studies in the area of the FBPP, it is assumed that the set of jobs assigned to a batch stick together during the whole process, i.e., the batch composition remains the same on all machines. Although, there are many real-world cases in which this assumption is valid, such as chemical and food industries (Damodaran and Srihari (2004)), but there are also real world cases that it is not necessary that the same batch composition should be considered during the whole process. Some examples of the FBPP with different batch compositions include the manufacturer of pastry, cakes, bakery, shoes, dyeing factory, and transportation systems. Apart from this, if the same batch composition is used on all machines, the machines with more capacity will work under their available capacity, since the batch composition is set based on the machine with the lowest capacity.

Motivated by industrial applications and considering the fact that we may obtain better solutions by allowing different batch compositions on different machines, we study this strategy and show how efficient it can be through our experiments. We show that the completion time of jobs is reduced by utilizing timely processed batches on machines. This being the case, the optimal solution obtained by traditional FBPP (following the same batch composition on machines) is either improved or not changed when different batch compositions are allowed. Another contribution of this paper is that there is only a few research available in the area of the FBPP which consider both the maximum number of jobs as well as the size capacity at the same time as the restrictions in the batch capacity.

The FBPP with minimization of makespan as the criterion is considered in this research. The processing time of each batch on each machine is equal to the maximum processing time of the jobs assigned to a batch (category C in Section 1.1). The batch capacity restricts both the maximum number of jobs as well as the size capacity at the same time (category 2 in Section 1.2). Each job can be assigned to different batches on different machines, led to different batch compositions on machines. To the best of our knowledge, there is no available research in the literature for this proposed research problem.

Usually, scheduling problems are described by a triple notation as $\alpha|\beta|\gamma$ for a better understanding. Unfortunately, even in the recently updated notations (Brucker (2007); Pinedo (2012)), the notations introduced for BPP are not comprehensive enough to cover all types of BPP. The rest of the paper is organized as follows. In Section 2, a MILP model is proposed for the research problem. A meta-heuristic algorithm based on particle swarm optimization (PSO) is proposed to heuristically

Table 1
A summary of batch processing problems in different shop environments.

Author(s)/shop environment	Batch composition		Batch processing time			Batch capacity restriction	
	Fixed	variable	Category A	Category B	Category C	Size capacity	# of jobs
Single machine							
Uzsoy (1994)	×				×		×
Damodaran et al. (2007)	×				×	×	×
Xu et al. (2012)	×				×	×	
Two-machine flowshop							
Damodaran and Srihari (Damodaran and Srihari, 2004)	×				×	×	
Liao and Liao (2008)	×				×	×	
Manjeshwar et al. (2009)	×				×	×	
Liao and Huang (2011)	×				×	×	
Tang and Liu (2009)	×				×	×	
Sung and Yoon (1997)	×		×			×	
Multi-machine flowshop							
Sung et al. (2000)	×		×				×
Matin et al. [This work]		×			×	×	×

solve this strongly NP-hard problem in Section 3. Two lower bounding mechanisms are proposed in Section 4 for the special cases of the research problem. Thereafter, in Section 5, the performance of the proposed PSO algorithm is evaluated by comparing the PSO solutions to the optimal solutions/lower bounds. Finally, a discussion and a concise conclusion along with potential future research are proposed in Sections 6 and 7, respectively.

2. The mathematical model

An MILP model is proposed to solve the proposed research problem optimally. The structure of the mathematical model developed in this research is different from the one developed by Damodaran et al. (2007) since this model considers multi-machine instead of single-machine and variable batch composition instead of fixed batch composition. We consider the problem of scheduling n jobs $N=\{1,2,\ldots,n\}$ on m machines $V=\{1,2,\ldots,m\}$ as b batches $B=\{1,2,\ldots,n\}$ to minimize makespan (C_{max}). The parameters, decision variables, and the model are as follows:

2.1. Parameters and indices

b The index used for the batches, $b \in B$
i The index used for the machines, $i \in V$
j The index used for the jobs, $j \in N$
s_j The size of job j, $\forall j \in N$
c_i The maximum number of jobs can be assigned to a batch on machine i, $\forall i \in V$
v_i The size capacity of any batch developed on machine i based on job sizes, $\forall i \in V$
p_{ij} The processing time of job j on machine i, $\forall i \in V, j \in N$
M A very large number

2.2. Decision variables

x_{ijb} 1; If job j is assigned to batch b on machine i; 0, otherwise. $\forall i \in V, j \in N, b \in B$
pb_{ib} The processing time of batch b on machine i, $\forall i \in V, b \in B$
$cjob_{ij}$ The completion time of job j on machine i, $\forall i \in V, j \in N$
cb_{ib} The completion time of batch b on machine i, $\forall i \in V, b \in B$
sb_{ib} The start time of processing batch b on machine i, $\forall i \in V, b \in B$
C_{max} Makespan of the schedule

2.3. Mixed-integer linear programing model (MILP)

$$\text{Minimize } C_{max} \tag{1}$$

$$\sum_{b\in B} x_{ijb} = 1, \quad \forall i \in V, j \in N \tag{2}$$

$$\sum_{j\in N} s_j x_{ijb} \le v_i, \quad \forall i \in V, b \in B \tag{3}$$

$$\sum_{j\in N} x_{ijb} \le c_i, \quad \forall i \in V, b \in B \tag{4}$$

$$pb_{ib} \ge p_{ij}{}^{*} x_{ijb}, \quad \forall i \in V, j \in N, b \in B \tag{5}$$

$$cb_{1b} = \sum_{k=1}^{b} pb_{1k}, \quad \forall b \in B \tag{6}$$

$$cb_{1b} = cb_{1(b-1)} + pb_{b1}, \quad \forall b \in B - \{1\} \tag{7}$$

$$cb_{ib} = sb_{ib} + pb_{ib}, \quad \forall i \in V, b \in B \tag{8}$$

$$sb_{11} = 0 \tag{9}$$

$$sb_{ib} \ge cjob_{(i-1)j} - M(1 - x_{ijb}), \quad \forall i \in V - \{1\}, j \in N, b \in B \tag{10}$$

$$sb_{ib} \ge cb_{i(b-1)}, \quad \forall i \in V, j \in N, b \in B - \{1\} \tag{11}$$

$$cjob_{ij} \ge cb_{ib} - M(1 - x_{ijb}), \quad \forall i \in V, j \in N, b \in B \tag{12}$$

$$cb_{ib} \ge cjob_{ij} - M(1 - x_{ijb}), \quad \forall i \in V, j \in N, b \in B \tag{13}$$

$$cb_{ib} - cb_{i(b-1)} \ge pb_{ib}, \quad \forall i \in V, b \in B - \{1\} \tag{14}$$

$$cjob_{ij} - cjob_{(i-1)j} \ge pb_{ib} - M(1 - x_{ijb}), \quad \forall i \in V - \{1\}, j \in N, b \in B \tag{15}$$

$$c_{max} \ge \sum_{b\in B} pb_{ib}, \quad \forall i \in V \tag{16}$$

$$c_{max} = cb_{mn} \tag{17}$$

$$x_{ijb} \in \{0,1\}, \quad \forall i \in V, j \in N, b \in B$$

$$pb_{ib},\ cb_{ib},\ cjob_{ij},\ sb_{ib} \ge 0, \quad \forall i \in V,\ j \in N,\ b \in B$$

The objective function is minimization of makespan as shown in Eq. (1). Constraint sets (2) through (4) are incorporated into the model to

determine the batch composition on each machine with respect to the batch capacity restrictions, i.e., both the number and total size of the jobs assigned to batches at the same time. Constraint set (2) ensures that each job is assigned to exactly one batch on each machine. Constraint set (3) is incorporated into the model to make sure that the total size of the jobs assigned to a batch on each machine is equal to or less than the size capacity of the batch on that machine. In addition, the number of jobs assigned to a batch on each machine should be equal to or less than the maximum number of jobs that can be assigned to the batch. Constraint set (4) is incorporated into the model for this reason. Constraint set (5) determines the processing time of each batch as the largest processing time of the jobs assigned to the batch on each machine. It is clear that for minimization of makespan criterion, all batches on the first machine are processed after each other with no delay. Thus, the completion time of each batch on the first machine, obtained by constraint sets (6) and (7), is equal to the completion time of the previous batch plus the processing time of that batch. Constraint set (8) is incorporated into the model to make sure that the completion time of each batch on each machine is equal to the time that the process of the batch is started on the machine plus the processing time of the batch.

Since all jobs and machines are available at the beginning of the planning horizon, the process of the first batch on the first machine starts at the beginning of the planning horizon. Constraint set (9) is incorporated into the model for this reason. The process of a batch on each machine can start if all jobs assigned to the batch are available to be processed by the machine. Thus, the process of all jobs assigned to a batch should be completed on the preceding machine. In order to support this fact, constraint set (10) is incorporated into the model. Also the process of each batch on each machine should start after the process of the preceding batch on the machine is completed. Constraint set (11) is incorporated into the model to support this fact. It is clear that the completion time of each job on each machine is equal to the completion time of the batch that the job belongs to on that machine. Constraint set (12) is incorporated into the model to support this fact. The completion time of each batch on each machine is also equal to or greater than the completion time of the jobs belonging to the batch as well. Constraint set (13) is incorporated into the model for this reason. Both constraint sets (12) and (13) express the fact that the completion time of a batch and the jobs belonging to that batch on each machine are equal.

The difference between the completion times of two consecutive batches on each machine should be equal to or greater than the processing time of the batch processed as the latest batch. Constraint set (14) is incorporated into the model to support this fact. The difference between the completion times of each job on two consecutive machines is at least equal to the processing time of the batch that the job belongs to on the second machine. Constraint set (15) is incorporated into the model to support this fact. Constraint set (16) is incorporated into the model since makespan is equal to or greater than the sum of all processing times of all batches on each machine. The completion time of the last batch on the last machine is equal to makespan. Constraint set (17) is incorporated into the model for this reason.

The proposed mathematical model can be easily adjusted to solve the problem with other objective functions such as minimization of total flow time and any objective function related to due dates with minor modifications. Liao and Liao (2008) proved that the two-machine flowshop BPP with minimization of makespan as the criterion is a NP-hard problem when the processing time of each batch is equal to the maximum processing time of the jobs assigned to the batch and each batch has only a restriction on the maximum number of jobs. Our proposed research problem is more complex than the one proposed by Liao and Liao (2008) because of our assumptions about the composition of batches regarding to the batch capacity restriction. Based on this insight, it is easy to see that the problem proposed in this research is easily reducible to the one already proven NP-hard. Thus, the fact that the proposed research problem is NP-hard, follows immediately.

3. Particle swarm optimization algorithm

As the proposed research problem is shown to be NP-hard, a meta-heuristic algorithm is required for solving industry-size problems. Since the feasible solution area of the proposed research problem is larger than regular flowshop scheduling problems (the jobs need to be assigned to different batches on different machines), population-based meta-heuristic algorithms might perform better than meta-heuristic algorithms based on a local search structure. Among the population-based meta-heuristic algorithms, PSO is chosen to heuristically solve the problem based on its good performance on flowshop scheduling problems (Liao et al. (2007); Tseng and Liao (2008); Hajinejad et al. (2011); Tadayon and Salmasi (2012)).

PSO is a population-based optimization algorithm proposed by Kennedy and Eberhart (Kennedy and Eberhart, 1995) for solving continuous optimization problems. It starts with several feasible solutions, named particles, as the first iteration. The number of initial feasible solutions is called P_{size}. Each particle, say the *ith* particle ($i = 1, 2, \ldots, P_{size}$), is presented by two *multi*-dimensional vectors as position, X_i, and velocity, V_i. All particles move toward areas with better objective function values by learning from the movement of swarm population. At each iteration, the position of a particle (X vector) is updated by calculating the velocity (V vector) using the differences between the current position of a particle and the two following positions:

- The position with the best objective function value observed ever by the *ith* particle at the *kth* iteration ($k = 1, 2, \ldots, Itr_{max}$), which is presented by *p-best*.
- The best position observed ever by all particles till the *kth* iteration, which is called *g-best*.

In general, in a PSO algorithm, the following equations are applied to update the position of particles during iterations:

$$V_i^k = w^*V_i^{k-1} + c1^*r1^*\left(pbest_i - X_i^k\right) + c2^*r2^*\left(gbest - X_i^k\right) \tag{18}$$

$$X_i^{k+1} = X_i^k + V_i^{k+1} \tag{19}$$

where:

V_i^k	The velocity of the *ith* particle in the *k*th iteration
X_i^k	The position of the *ith* particle in the *kth* iteration
$pbest_i$	The vector for the best known position of the *ith* particle
gbest	The best position vector of all particles in the population
w	The inertia weight of a particle on the next iteration
$c1$ and $c2$	The acceleration coefficients
$r1$ and $r2$	Random numbers between 0 and 1

The constant values of $c1$ and $c2$, known as acceleration coefficients, determine the impact of $pbest_i$ and *gbest* to define velocity, respectively, while $r1$ and $r2$ are incorporated in velocity to consider uncertainty in the meta-heuristic algorithm. w is the inertia weight related to the *k*th iteration, which determines the impact of the previous velocity of the particle on the next iteration.

The PSO algorithm might be trapped in local optima. Therefore, the following mechanism is applied within the structure of the algorithm to reduce the chance of converging into local optima. In order to improve the performance of the PSO algorithm, a multiplier χ in Eq. (18) can be implemented to accelerate the process of the convergence (Poli et al. (2007)) as follows:

$$V_i^k \leftarrow \chi\left(w^*V_i^{k-1} + c1^*r1^*\left(pbest_i - X_i^k\right) + c2^*r2^*\left(gbest - X_i^k\right)\right) \tag{20}$$

In order to control the extreme roaming of particles outside of the search space, the new velocity and position values are restricted to the interval $[V^{min}, V^{max}]$ and $[X^{min}, X^{max}]$, respectively. In the $Itr_{Max}{}^{th}$ iteration of PSO, *gbest* is reported. In the following, the details of the proposed PSO algorithm for solving the research problem such as generating the initial

population, solution representation, decoding, encoding, and updating the particles, incorporating a suitable local search structure, and diversifying technique are explained.

3.1. Initial population

Since the feasible solution area of the proposed research problem is more diverse than regular flowshop scheduling problems, different rules are used to generate the initial feasible solutions as the members of the initial population in order to have more diversity. These rules are as follows:

1 ***NEH Single Job Assignment approach*:** The first particle of the population is generated based on a sequence of processing the jobs by using Nawaz et al. (1983) algorithm which is known as NEH algorithm in the scheduling literature. They proposed a heuristic algorithm for solving flowshop problems with the minimization of makespan. In order to use this algorithm to generate an initial solution, we assume that each machine can process at most one job at the same time. Then, the sequence of jobs generated by the NEH algorithm is used to assign the jobs to the batches. In order to generate this initial solution, we assume that the batch capacity restricting both the number and total size of assigned jobs is the same on all machines and is equal to the minimum value of all possible batch capacities on all machines. For this initial solution, we assume that the batch composition remains the same on all machines.

2 ***NEH Batch Assignment approach*:** Without loss of generality, assume that $j_{i_1}, j_{i_2}, \ldots, j_{i_m}$ is the sequence of jobs on machines, where $i_k \in \{1, \ldots, m\}$ and j_{i_k} is the sequence of jobs generated by the NEH algorithm on machine k. To assign the jobs to batches, use the first fit heuristic on each machine, proposed by Uzsoy (1994). In this heuristic, the first job in the sequence j_{i_1} is assigned to the first batch that can accommodate this job, i.e., the batch capacity restrictions, including the number of assigned jobs and the size capacity are satisfied (in this case, it is b_1). Then, the second job in the sequence j_{i_1} will be assigned to the first batch that satisfies the batch capacity restrictions (that is, j_2 is assigned to batch b_1 if this batch can accommodate this job; otherwise, it is assigned to batch b_2). Following the same procedure, the rest of jobs are assigned to batches on the first machine and other machines. Contrary to previous initial solution mechanism, the batch capacity restrictions correspond to the machine assigned to the batch so that the batch compositions on machines can be different.

3 ***CDS Single Jobs Assignment Approach*:** The initial solution of the third particle is generated based on a sequence of processing the jobs by using Campbell et al. (1970) algorithm which is known as CDS algorithm in the scheduling literature. They proposed a heuristic algorithm for solving flowshop problems with the minimization of makespan. Similar to the first initial solution mechanism, batch sequence is determined on machines with the help of the sequence of jobs generated based on the CDS algorithm. In addition, the batch composition remains the same on all machines.

4 ***CDS Batch Assignment Approach*:** Following the same procedure implemented for the second initial solution mechanism, batch sequence on machines is determined by the CDS algorithm and the first fit heuristic to arrange the jobs on each machine and, consequently, assign the jobs to batches on each machine, respectively. This initial solution mechanism might generate different batch compositions on machines.

5 ***The Longest Processing Time First Approach (LPT):*** Assume that the restrictions on the batch capacity is the same on all machines and is equal to the minimum value of all possible batch capacities on all machines. In the LPT heuristic, sort the jobs based on their *sum of processing time of each job on all machines* in a descending order. Without loss of generality, assume that $j_1, j_2, \ldots, j_n$ is the order of jobs after sorting. On the first machine, assign j_1 to b_1 (the first batch). Then, assign j_2 to b_1 if the batch capacity restrictions are satisfied; otherwise, assign j_2 to b_2 and similarly for the rest of jobs. Following the same batch composition on other machines, batch sequence is determined on machines.

6 ***The First Fit Longest Processing Time First (FFLPT) Heuristic:*** Similar to the LPT heuristic, in the FFLPT heuristic, sort the jobs in a descending order based on their *sum of the processing time of each job on all machines.* Then, assign the jobs to batches by applying the first fit heuristic on machines with respect to corresponding batch capacity restrictions on each machine. Therefore, unlike the LPT heuristic, the batch compositions on machines can be different.

7 ***The Shortest Processing Time First (SPT) Approach*:** In the SPT heuristic, sort the jobs based on their *sum of the processing times of each job on all machines* in an increasing order. Similar to the LPT heuristic, assign the jobs to batches on the first machine and follow the same batch composition for the rest of machines.

8 ***The First Fit Smallest Processing Time First (FFSPT) Approach*:** Similar to the FFLPT heuristic, determine different batch compositions and sequences on machine with respect to an increasing order of the jobs at the beginning.

9 ***Random Solutions*:** The rest of the members of the initial population, i.e. $(P_{size} - 8)$ are generated randomly. First, sort the jobs on all machines randomly. Then, assign the jobs to batches with respect to the first fit heuristic and the batch capacity restrictions. As a result, different batch compositions on machines can be obtained.

3.2. Solution representation for PSO

By recognizing that x_{ij}^k represents the assigned batch to job j $(j = 1, 2, \ldots, n)$ on the first machine related to the *ith* particle $(i = 1, 2, \ldots, P_{size})$ in the *kth* iteration $(k = 1, 2, \ldots, Itr_{max})$ of PSO, $X_i^k = [x_{i1}^k, \ldots, x_{ij}^k, \ldots, x_{in}^k]$ represents the assigned batches to jobs on the first machine for the *kth* iteration of the *ith* particle. For instance, $x_{12}^5 = 3$ indicates that in the fifth iteration of PSO, the second job of the first particle is assigned to batch number 3 on the first machine. The initial solution mechanisms developed in Section 3.1 determine $X_i^1, \forall i = 1, 2, \ldots, P_{size}$.

3.3. Decoding & updating particles

In order to construct a new particle, Eq. (19) is applied to update the position vector. However, by using the classical method, the outcome, i.e., X_i^{k+1} cannot be directly considered as a feasible solution. Thus, we propose several modifications on the new generated X_i^{k+1} to interpret it as the vector of the new position of the particle. Let's consider the outcome of Eq. (19) as Z_i^k:

$$Z_i^k = X_i^k + V_i^{K+1} \tag{21}$$

In spite of the original PSO algorithm, since the addressed research problem belongs to a discrete space, the updated particle i.e., Z_i^k, should be modified to be considered as a feasible solution (i.e., $X_i^{k+1} \leftarrow Z_i^k$). The following two-step modification process is proposed to determine job assignment to batches and batch sequence on machines.

3.3.1. A modification process on the first machine

In the flowshop scheduling problems, the downstream machines can start to process the jobs when the processing of the jobs on upstream machines is completed. Thus, a job assignment to batches along with a batch sequence on upstream machines *quicker* may provide a better solution by reducing the idle times on downstream machines. This idea is applied to the first step in the modification process. The approach applied for this step is based on the job tendency to be settled in a batch which is similar to the approach proposed by Damodaran and Srihari (2004). In

this approach, after applying Eq. (21) to the vector X_i^k, the corresponding members of the vector Z_i^k are sorted in an ascending order. It is assumed that the lower values indicate the tendency of a job to be settled in early batches. Then, the jobs with the lowest and the highest orders are paired and assigned to the same batch. With respect to the batch capacity restrictions on the first machine and following the same approach (i.e., for the jobs with the second lowest and the second highest orders, and so on), the batch composition on the first machine (i.e., bottleneck machine) is determined.

3.3.2. A modification process on the other machines

In the second step of the modification process, the batch compositions on the rest of the machines (the machines other than the bottleneck machine) are determined by using either the *First Fit Assignment* (FFA) or the *Best Fit Assignment* (BFA) approaches. In the following, the FFA and the BFA approaches are explained.

3.3.2.1. The FFA approach. In this approach, job $j \in N$ should be assigned to the first available batch $b \in B$ that can accommodate this job with respect to the batch capacity restrictions. Let x_i^k be the solution corresponding to k^{th} iteration of particle i on the first machine. Assume job j is assigned to batch $b \in B$ on machine $i \in V$. In the FFA approach, job j on machine $i+1$ ($i \leq m-1$) is assigned to batch $b' \in B$ if:

- $b' \geq b$, and
- batch b' satisfies the batch capacity restrictions.

3.3.2.2. The BFA approach. In this approach, job $j \in J$ should be assigned to the best available batch $b \in B$ that can accommodate this job with respect to the batch capacity restrictions. The best available batch b on a machine is a batch with the minimum number of assigned jobs and the largest available size capacity between developed batches on the machine so that the batch capacity restrictions are not violated by assigning job j to batch b. If the best available batch is not obtainable due to any violation of the batch capacity restrictions, job j is assigned to a new batch. Therefore, similar to the FFA approach, job j on machine $i+1$ is assigned to batch $b\prime$ if:

- $b\prime \geq b$,
- $b\prime = argmin_{b \in B} S_b$, and
- Batch b' satisfies the batch capacity restrictions.

where S_b denotes the available size capacity of batch b ($\forall b \in B$) developed on machine $i+1$ ($i \leq m-1$).

In the case that for two consecutive machines i and $i+1$, the size capacity of the later machine is larger than the first one, both the FFA and the BFA approaches usually follow the same batch composition on both machines. However, the batch composition might be different if the preceding machine has more capacity.

3.3.2.3. Remark. In both the FFA and the BFA approaches, one of the conditions for job assignment to batches is that $b' \geq b$. Therefore, job assignment to batches might be the same due to similarity in the assignment approach. For this reason, and also for more diversity in neighborhood policies, the restriction $b' \geq b$ is relaxed and substituted by $b' \geq 0$ for the BFA approach. Consequently, different job assignments to batches, i.e., different batch compositions, are developed on machines.

3.4. Improving the performance of PSO by a local search structure

Previous researches on implementing the PSO algorithms in flowshop scheduling problems (Hajinejad et al. (2011); Tadayon and Salmasi (2012); Arabameri and Salmasi (2013)) show that the quality of the solution obtained by a basic PSO algorithm can be significantly improved when it is accompanied by a local search structure. This is our motivation to propose several neighborhood policies to find better solutions after updating each particle. Therefore, a local search structure is incorporated into the PSO algorithm to find several neighbor solutions of X_i^k. If a neighbor solution with a better objective function is found, the particle is then substituted by its best neighbor solution.

3.4.1. Neighborhood mechanisms

Our preliminary investigations on a set of experiments indicate that a swap-related operator generates better solutions compared to an insertion-related operator. The reason lies in the fact that an insertion of a job into a batch might violate the batch capacity, particularly the size capacity, since the job assignment to batches are determined so that the total size of jobs assigned to batches are in the same range. For the same reason, a swap-related operator is capable of generating good solutions compared to an insertion-related operator. Apart from this, a combination of both operators not only increases the computational time of the search algorithm, but also does not improve the quality of the final solution significantly.

A swap-related operator is performed on job assignment within batches as well as batch sequence on a particular machine of X_i^k to generate neighbor solutions. If a neighborhood mechanism is applied for machine i ($i \in V$), the assignment of jobs to batches for the following machines ($i+1, \ldots, m$) are determined with the help of the FFA and/or the BFA approaches (Section 3.3.2). In the following, two neighborhood mechanisms in terms of a swap-related operator are explained.

i. ***Changing job assignment within batches*:** In this neighborhood mechanism, two jobs belonging to two different batches on a particular machine are swapped if interchanging these jobs does not violate the size capacity on the machine. It is worth noting that in this neighborhood mechanism, job j and j', belonging to batches b and b' on the same machine, respectively, will be swapped only when the processing time of job j and j' are equal to or smaller than the processing time of batch b' and b, respectively.
ii. ***Changing batch sequence on machines*:** Similar to the above neighborhood mechanism, the batches including one or more jobs are swapped on a particular machine. Let assume that J_i and J_t are the set of jobs corresponding to batches b_i and b_t on the same machine, respectively. These two batches swap on a the machine, that is, all of the jobs in b_i will be located at b_t and vice versa, if at least one of these two batches has not been swapped yet.

3.5. Encoding particles

After finishing any local search structure implemented at the end of the *kth* iteration of PSO and before starting the $(k+1)^{th}$ iteration of PSO, an encoding of particles is required to convert members of X_i^{k+1} to continues values. If $x_{ij}^{k+1} = b$ ($b \in B$), then x_{ij}^{k+1} is replaced by a random number generated in $(b, b+1]$.

3.6. Proposed PSO algorithms

Three different PSO algorithms are proposed based on different local search structures. In all of these local search structures, the same neighborhood mechanisms are implemented. Apart from this, the assignment of jobs to batches on the first machine is determined with the help of the first step of the modification process (Section 3.3.1), while for the assignment of jobs to batches on other machines, the BFA and/or the FFA approach is applied (Section 3.3.2). Local search structures are as follows:

a. ***Best Neighbor (BN) local search:*** In this mechanism, for each particle, the assignment of jobs to batches on machine $i \in V - \{1\}$ are determined by applying both the FFA and the BFA approaches, and

consequently, in each iteration, the best neighbor solution is considered as X_i^{k+1}.

b. ***Best Fit (BF) local search:*** In this mechanism, for each particle, the assignment of jobs to batches on machine $i \in V - \{1\}$ are determined by applying only the BFA approach and, consequently, in each iteration, the best neighbor solution is considered as X_i^{k+1}.

c. ***First Fit (FF) local search:*** In this mechanism, for each particle, the assignment of jobs to batches on machine $i \in V - \{1\}$ are determined by applying only the FFA approach and, consequently, in each iteration, the best neighbor solution is considered as X_i^{k+1}.

3.7. Diversifying technique

The position of the best known particle in the population has a significant effect on determining the velocity and direction of particle movements. If the best position does not change in several consecutive iterations, it is possible that all particles converge around this position. In this case, several particles within the population may take the value of *gbest* and stop searching in the solution space. Therefore, the diversity of particles in the population is decreased considerably and the search power of the algorithm is weakened. A diversifying technique is applied in the structure of PSO, at every 20 iterations in order to resolve this issue. This procedure examines all particles within the population to find any two identical particles and substitutes the iterant particle by a random valued vector.

4. Generating a lower bounding method for the research problem

Since the proposed research problem is shown to be NP-hard, it is required to generate good quality lower bounds to evaluate the performance of the proposed meta-heuristic algorithms, particularly for large-size problems. The possibility of existing lower bounding methods for the proposed research problem is discussed in three different scenarios as follows:

Scenario 1. Each batch has both the maximum number of assigned jobs and the size capacity restrictions on the batch capacity

In this scenario, since the jobs are assigned to batches based on two different restrictions, it is not possible to propose a lower bounding method based on the scheduling characteristics of the problem. In such problems, using general optimization techniques such as lagrangian relaxation or branch-and-price algorithm might be the only possible methods to generate efficient lower bounds for the research problem. The complexity of the research problem is decreased if the jobs are assigned to batches based on only one of the aspects of the batch capacity restrictions. The two following scenarios are presented based on this fact.

Scenario 2. Each batch has only the maximum number of assigned jobs restriction on the batch capacity.

By sorting the jobs on each machine based on the LPT rule and, consequently, developing batches with respect to only the maximum number of assigned jobs restriction, lower bounds are generated as follows:

***Step* 1:** Set $i = 1$
***Step* 2:** Order the jobs on machine i according to the LPT rule. With respect to this order, assign the first c_i jobs to the first batch. Following the same procedure, assign the rest of jobs to other batches. The last batch might have fewer jobs than c_i.
***Step* 3:** Calculate makespan of processing batches on machine i and save it as LB_i.
***Step* 4:** Find the job that has the minimum sum of processing time on all machines except machine i. Add the processing time of that job to LB_i.
***Step* 5:** If $i < m$, set $i = i + 1$ and go to step 2; otherwise, go to step 6.
***Step* 6:** The lower bound (LB) of the problem is calculated by the following formula: $LB = \max\{LB_1, LB_2 \ldots LB_m\}$.

Scenario 3. Each batch has only the size capacity restriction on the batch capacity

Liao and Liao (2008) proposed a lower bounding method for the two machine BPP with the same batch composition on both machines. They assumed that the jobs can be split into two parts and each part can be processed in a different batch. We generalize this method for the *multi-machine* FBPP by considering different batch compositions on machines. The steps of this lower bounding method are the same as the second scenario except for step 2 which is determined as follows:

Order the jobs on machine i according to the LPT rule. Assign the jobs to batches based on this order with respect to the size capacity on machine i (v_i) and the size of jobs assigned to batches. Assume that the jobs can be split into two parts so that those parts can be processed in two consecutive batches. In this case, the processing time of the split job is considered the same as its processing time in both batches.

5. Computational experiments

The computational experiments are performed to compare the performance of the three proposed PSO algorithms. First, the performance of the proposed PSO algorithms are compared with each other at three levels related to the number of machines. Then, the performance of the best solutions obtained by all PSO algorithms is evaluated by the optimal solutions/lower bounds. For this purpose, the proposed PSO algorithms are implemented by using Java programming language, while the optimal solutions/lower bounds are obtained from solving the MILP models with CPLEX 12.2 (IBM, 2009). All runs have been performed on identical computers with Intel(R) Core(TM) i5-52000U CPU @ 2.20 GHz processors & 8.00 GB of RAM.

5.1. Data generation

In order to evaluate the performance of the proposed meta-heuristic algorithms, several test problems are randomly generated. The complexity of problems increase as the number of jobs and machines along with the number of developed batches on machines are increased. Therefore, the computational time of CPLEX increases exponentially as the size of problems is increased. This being the case, test problems specifications are determined as follows:

- *Problem size*: Three different sizes of problems are considered based on the number of jobs as small-, medium-, and large-size problems.
- *Number of* jobs: The number of jobs are generated from uniform distributions in $n = unif[5, 20]$, $unif[21, 50]$, and $unif[51, 100]$, for small-, medium-, and large-size problems, respectively.
- *Number of machines*: For each size of problem, the flowshop environment is considered with respect to three levels for the number of machines, i.e., two-, three-, and six-machine flowshop problem.
- *Job run time*: Since the combination and assignment of jobs to batches, i.e., batch composition, are very sensitive to the processing time of jobs, two ranges of processing times are considered and randomly generated for each machine. The processing time of jobs are generated from uniform distribution in $P_{ij} = unif[1, 10]$ and $unif[1, 20]$.
- *Job size*: Three levels of job sizes are generated from uniform distribution in $s_j = unif[2, 4]$ and $unif[5, 8]$ for small and large jobs, respectively, as well as $s_j = unif[1, 10]$ for a combination of small and large jobs (Damodaran et al. (2006)).
- *Number of jobs* per *batch*: The maximum number of jobs per batch is considered 5, 10, and 15, for small-, medium-, and large-size

problems, respectively. Consequently, the number of developed batches on each machine is in the range of [1, 4], [2, 5], and [4, 7] batches for small-, medium-, and large-size problems, respectively, irrespective of the batch capacities.

The emphasis in this research is on the case that the size of the jobs is large relative to the capacity of batches; otherwise, usually most of the algorithms result similar batch compositions on machines. If this were the case, regardless of the algorithm that we apply, usually one batch can accommodate a large-number of jobs. Therefore, the chance that many of the algorithms generate similar batch composition is high. On the other hand, there should be a balance between the size capacity per batch, v_i, the maximum number of jobs per batch, c_i, and job sizes, s_j, so that $v_i = c_i \times \bar{s}_j$, where $\bar{s}_j$ represents the average job size, i.e., $\bar{s}_j = (\sum_{j=1}^{n} s_j)/n$. Therefore, for the i^{th} machine, v_i is determined after generating the job sizes as follows:

- *The size capacity*: With respect to $s_j = unif[1, 10]$ which covers all job sizes, v_i will be generated in [1, 50], [1, 100], and [1, 150], for small-, medium-, and large-size problems, respectively.

5.2. Parameter tuning

The best estimate for the PSO parameters are calculated by experimental design techniques. Based on extensive experiments using test problems, different from the ones used in algorithms comparison, the PSO parameters are tuned (Appendix A). During the evolution process of PSO, the non-linear dynamic coefficients are considered at each iteration as follows:

$$c1 = c1^{min} + (c1^{max} - c1^{min})/(1 + \exp(-a(Itr_{max} - k)/Itr_{max})) \quad (22)$$

$$c2 = c2^{min} + (c2^{max} - c2^{min})/(1 + \exp(-ak/Itr_{max})) \quad (23)$$

$$\omega = \omega^{min} + (\omega^{max} - \omega^{min})/(1 + \exp(-a(Itr_{max} - t)/Itr_{max})) \quad (24)$$

$w^{max}/c1^{max}/c2^{max}$ represents the highest value of $w/c1/c2$ and $w^{min}/c1^{min}/c2^{min}$ represents the lowest value of $w/c1/c2$. Parameter a, decremental factor, has a constant value. The value of multiplier χ is calculated as $\chi = \frac{2}{C-2+\sqrt{C^2-4C}}$, where $c = c1 + c2$, and $c1 + c2 > 4$. In most research, the values assigned to parameters $c1$ and $c2$ are approximately close to 2. If the multiplier χ is used, c is set to 4.1, $c1 = c2$, and consequently, the constant multiplier χ is approximately 0.7298; otherwise, the value of $c1$ and $c2$ are determined based on non-linear equations presented by Eqs. (22) and (23).

Since $P_{size} = 20$, 30, and 50 for small-, medium-, and large-size problems, respectively, we will generate 8 initial solutions with the help of initial solution mechanisms developed in Section 3.1 along with $P_{size} - 8$ different random initial solutions for each problem size. Itr_{max} is the maximum number of iterations of each algorithm, while Itr_{max}^{PSO} is the maximum number of iterations for each local search structure in all proposed PSO algorithms, which is equal to $0.20n$, $0.15n$, and $0.10n$, for small-, medium-, and large-size problems.

5.3. Experimental results

A split-plot design (Montgomery, 2009) is implemented with five factors, shown in Table 2. Five replications have randomly been generated for any combination of the problem size (*PS*), processing time (*PT*), job size (*JS*), and number of machines (*MN*) factors. Each of these replications has been solved by all three algorithms (*Alg*) in a random order, which resulted in a total number of 810 computer runs (54 combination of *PS*, *PT*, *JS*, and *MN* × 5 replications × 3 *Alg* = 810). The statistical model for this design is:

Table 2
Factors and their levels in the experiment.

Factor name		Levels
Whole-plot		
Problem size	*PS*	Small – size, Medium – size, and Large – size
Processing time	*PT*	[1, 10] and [1, 20]
Job size	*JS*	[2, 4], [5, 8], and [1, 10]
Number of machines	*MN*	2, 3, and 6
Sub-plot		
Algorithm	*Alg*	*Alg*1, *Alg*2, and *Alg*3

$$\begin{aligned} y_{ijklmn} = {} & \mu + \tau_i + \vartheta_j + \rho_k + \gamma_l + \delta_m + (\vartheta\rho)_{jk} + (\vartheta\gamma)_{jl} + (\vartheta\delta)_{jm} + (\rho\gamma)_{kl} \\ & + (\rho\delta)_{km} + (\gamma\delta)_{lm} + (\vartheta\rho\gamma)_{jkl} + (\vartheta\rho\delta)_{jkm} + (\rho\gamma\delta)_{klm} + (\vartheta\rho\gamma\delta)_{jklm} \\ & + \theta_{ijklm} + \varphi_n + (\vartheta\varphi)_{jn} + (\rho\varphi)_{kn} + (\gamma\varphi)_{ln} + (\delta\varphi)_{mn} + (\vartheta\rho\varphi)_{jkn} \\ & + (\vartheta\gamma\varphi)_{jln} + (\vartheta\delta\varphi)_{jmn} + (\rho\gamma\varphi)_{kln} + (\rho\delta\varphi)_{kmn} + (\gamma\delta\varphi)_{lmn} \\ & + (\vartheta\rho\gamma\varphi)_{jkln} + (\vartheta\rho\delta\varphi)_{jkmn} + (\rho\gamma\delta\varphi)_{klmn} + (\vartheta\rho\gamma\delta\varphi)_{jklmn} \end{aligned}$$

$i = 1, 2, 3, 4, 5$; $j, l, m = 1, 2, 3$; $k = 1, 2$; and $n = 1, 2, 3$ where μ is the overall mean effect, τ_i is the replicate effect, ϑ_j is the effect of j^{th} level of *PS*, ρ_k is the effect of k^{th} level of *PT*, γ_l is the effect of l^{th} level of *JS*, δ_m is the effect of m^{th} level of *MN*, and finally φ_n shows the effect of n^{th} level of *Alg*. Due to violations in normality assumption and constant variance assumption, the natural logarithm transformation of the response variable has been employed to totally resolve all the deviations (R 2. 13.0. (RStudio, 2011)). The resulting ANOVA table is presented in Appendix B.

In the whole-plot, there are convincing evidences of non-zero differences between different levels of *PS*, *PT*, and *JS* ($P_{value} < 0.00001$) and a strong evidence of non-zero differences between different levels of *MN*. The effect of all other interactions in the whole plot are not significant. There is a convincing evidence of a nonzero difference between different algorithms ($P_{value} < 0.00001$). There is a moderate and strong evidence of a nonzero difference between the interaction effect of *PT:Alg*, *PS:Alg*, and *MN:Alg*, respectively. Since the interpretation of a high rank interaction is quite difficult to explain, only the interactions between *PT:Alg*, *PS:Alg*, and *MN:Alg* are briefly explained with the help of following sensitivity analyses which are performed to compare the performance of developed algorithms at each level of *PT*, *PS*, and *MN*.

5.3.1. The comparison between developed algorithms based on Problem size

The principal result of a paired *t*-test performed to compare the different levels ($3 \times 3 = 9$) of *PS:Alg* are shown in Table 3. Based on P_{value} at significant level of 5% for each comparison in each problem set (small, medium, and large), it can be concluded that the problem size has an effect on the performance of developed algorithms. Since the average objective function value of the PSO algorithm with the best neighbor local search (i.e., PSO_{BN}) is less than PSO algorithms with both the best fit and first fit neighbor local search (i.e., PSO_{BF} and PSO_{FF}, respectively) for all three-problem sets, it can be concluded that PSO_{BN} provides better solutions for the proposed research problem, particularly for medium and large-size problems. Although PSO_{BF} presents the same performance compared to PSO_{FF} for large-size problems, it presents a slightly better performance compared to PSO_{FF} for small-size problems.

5.3.2. The comparison between developed algorithms based on processing time

The principal result of a paired *t*-test performed to compare the different levels ($2 \times 3 = 6$) of *PT:Alg* are shown in Table 4. Based on P_{value} at significant level of 5% for each comparison in each problem set ([1, 10] and [1, 20] processing time of jobs), it can be concluded that there is a significant difference between the results of these algorithms in each level of processing time. Since the average objective function value of PSO_{BN} is less than PSO_{BF} and PSO_{FF}, for all two-problem sets, it can be concluded that PSO_{BN} provides better solutions for the proposed research problem, particularly when $P_{ij} = unif[1, 20]$. Although PSO_{BF} presents the

Table 3
Paired *t*-test for two-, three-, and six-machine problems.

PS	Pair	Paired differences					*t*	*df*	Sig. (2-tailed)
		Mean	Std. deviation	Std. error mean	95% confidence interval of the difference				
					Lower	Upper			
Small	$PSO_{BN} - PSO_{BF}$	9.957	112.273	20.498	−32.946	52.859	0.486	29	**6.30805E-01**
	$PSO_{BN} - PSO_{FF}$	63.539	118.426	21.621	18.285	108.793	2.939	29	**6.40432E-03**
	$PSO_{BF} - PSO_{FF}$	53.582	112.671	20.571	10.528	96.637	2.605	29	**1.43523E-02**
Medium	$PSO_{BN} - PSO_{BF}$	44.664	135.015	24.650	−6.929	96.257	1.812	29	**8.03704E-02**
	$PSO_{BN} - PSO_{FF}$	90.034	164.120	29.964	27.319	152.749	3.005	29	**5.43452E-03**
	$PSO_{BF} - PSO_{FF}$	45.371	208.651	38.094	−34.361	125.102	1.191	29	**2.43308E-01**
Large	$PSO_{BN} - PSO_{BF}$	44.050	187.750	34.278	−27.694	115.795	1.285	29	**2.08935E-01**
	$PSO_{BN} - PSO_{FF}$	80.584	189.826	34.657	8.046	153.122	2.325	29	**2.72649E-02**
	$PSO_{BF} - PSO_{FF}$	36.534	161.088	29.410	−25.022	98.090	1.242	29	**2.24112E-01**

Table 4
Paired *t*-test for problems with [1, 10] and [1, 20] processing time.

PT	Pair	Paired differences					*t*	*df*	Sig. (2-tailed)
		Mean	Std. deviation	Std. error mean	95% confidence interval of the difference				
					Lower	Upper			
[1,10]	$PSO_{BN} - PSO_{BF}$	133.363	462.379	84.419	−43.325	310.051	1.580	29	**1.25004E-01**
	$PSO_{BN} - PSO_{FF}$	198.704	648.729	118.441	−49.193	446.601	1.678	29	**1.04161E-01**
	$PSO_{BF} - PSO_{FF}$	65.341	593.390	108.338	−161.410	292.092	0.603	29	**5.51114E-01**
[1,20]	$PSO_{BN} - PSO_{BF}$	179.479	842.025	153.732	−142.282	501.240	1.167	29	**2.52526E-01**
	$PSO_{BN} - PSO_{FF}$	399.753	943.316	172.225	39.285	760.220	2.321	29	**2.75135E-02**
	$PSO_{BF} - PSO_{FF}$	220.273	721.015	131.639	−55.247	495.793	1.673	29	**1.05020E-01**

same performance compared to PSO_{FF} when $P_{ij} = unif[1, 10]$, it presents a better performance compared to PSO_{FF} for the second set of problems, i.e., $P_{ij} = unif[1, 20]$. With respect to the interaction effect of *PT:Alg*, it can be concluded that the batch composition (the combinations and assignment of jobs to batches) are very sensitive to the processing time of jobs.

5.3.3. The comparison between developed algorithms based on number of machines

The principal result of a paired *t*-test performed to compare the different levels ($3 \times 3 = 9$) of *MN:Alg* are shown in Table 5. Based on P_{value} at significant level of 5% for each comparison in each problem set (two, three, and six machines problems), it can be concluded that there is a significant difference between the results of these algorithms in each level of number of machines. Since the average objective function value of PSO_{BN} is less than PSO_{BF} and PSO_{FF} for all three-problem sets, it can be concluded that PSO_{BN} provides better solutions for the proposed research problem. Likewise, PSO_{BF} presents a better performance compared to PSO_{FF} for all problem sets. The difference between each pair of the PSO algorithms are shown by box plots depicted in Appendix C, at two-, three-, and six-machine problems.

5.4. Algorithm performance

It is worth comparing the performance of the proposed PSO algorithms with respect to the optimal solutions/lower bounds of the MILP model. Since the proposed problem is NP-hard, the optimal solution or good quality lower bounds cannot be found in a polynomial time. However, for fairly small-size problems, this non-polynomial time is still affordable, which gives an opportunity to measure the quality of the PSO algorithms. The results of these runs are summarized in Table 6. The preliminary experiments show that the optimal solutions and good quality lower bounds are achievable by CPLEX, for fairly small-size and small-size problems, respectively. Therefore, the result of three proposed PSO algorithms are compared to the result of CPLEX. The best result of lower bounds obtained by CPLEX along with the lower bounds proposed by the second and the third scenarios related to lower bounding mechanisms is considered as the best lower bound for each test problem.

The deviation of the best solution, found by the PSO algorithms (UB_{PSO}), from the optimal solutions/upper bounds, obtained by CPLEX

Table 5
Paired *t*-test for two-, three-, and six-machine problems.

M|Cs	Pair	Paired differences					*t*	*df*	Sig. (2-tailed)
		Mean	Std. deviation	Std. error mean	95% confidence interval of the difference				
					Lower	Upper			
2	$PSO_{BN} - PSO_{BF}$	19.072	29.703	5.423	7.721	30.422	3.517	29	**1.45800E-03**
	$PSO_{BN} - PSO_{FF}$	47.863	46.919	8.566	29.934	65.792	5.587	29	**1.00000E-05**
	$PSO_{BF} - PSO_{FF}$	28.792	48.921	8.932	10.098	47.486	3.224	29	**3.12100E-03**
3	$PSO_{BN} - PSO_{BF}$	22.804	37.138	6.780	8.612	36.995	3.363	29	**2.18100E-03**
	$PSO_{BN} - PSO_{FF}$	79.985	57.852	10.562	57.878	102.091	7.573	29	**1.00000E-05**
	$PSO_{BF} - PSO_{FF}$	57.181	62.252	11.366	33.393	80.969	5.031	29	**2.30000E-05**
6	$PSO_{BN} - PSO_{BF}$	34.417	53.039	9.683	14.150	54.685	3.554	29	**1.32200E-03**
	$PSO_{BN} - PSO_{FF}$	110.574	78.295	14.295	80.655	140.493	7.735	29	**1.00000E-05**
	$PSO_{BF} - PSO_{FF}$	76.157	86.721	15.833	43.018	109.295	4.810	29	**4.30000E-05**

Table 6
Performance of the PSO algorithms compared to the optimal solutions/lower bounds.

# of m|c	PSO_{BN}		PSO_{BF}		PSO_{FF}		$CPLEX$			Lower Bounding Mechanism		Percentage Deviation		
	UB_{BN}	CT_{BN}	UB_{BF}	CT_{BF}	UB_{FF}	CT_{FF}	UB_{CPLEX}	LB_{CPLEX}	CT_{CPLEX}	LB_2	LB_3	UB_{CPLEX} vs UB_{PSO}	UB_{CPLEX} vs LB_{Best}	UB_{PSO} vs LB_{Best}
2	91	214	99	117	107	53	89	89	8065	74	75	2.1%	0.0%	2.1%
	121	562	125	375	130	310	121	121	4484	108	101	0.0%	0.0%	0.0%
	73	616	76	81	77	74	73	73	7148	63	66	0.0%	0.0%	0.0%
	87	571	84	364	89	103	84	84	4619	73	71	0.0%	0.0%	0.0%
	100	200	104	211	104	237	97	97	11,749	87	86	2.7%	0.0%	2.7%
	63	200	63	107	72	67	63	63	10,669	50	63	0.0%	0.0%	0.0%
	82	203	82	102	91	241	82	82	7395	66	76	0.0%	0.0%	0.0%
	129	335	141	104	135	319	129	129	14,941	109	119	0.0%	0.0%	0.0%
	117	233	117	303	128	149	117	117	14,071	98	100	0.0%	0.0%	0.0%
	121	592	129	115	132	186	121	121	4993	113	105	0.0%	0.0%	0.0%
3	108	612	108	423	111	337	112	108	13,814	93	103	−3.6%	3.7%	0.0%
	102	654	105	329	116	220	102	102	15,859	86	88	0.0%	0.0%	0.0%
	70	493	74	426	73	456	70	67	21,972	54	58	0.0%	4.6%	4.6%
	77	523	75	486	82	378	78	73	23,957	68	61	−3.8%	7.1%	3.0%
	130	779	138	390	141	385	136	118	24,235	98	127	−4.2%	7.1%	2.6%
	100	474	96	317	95	161	98	92	27,084	80	85	−3.1%	6.3%	3.1%
	90	378	89	214	88	286	91	88	13,127	78	75	−3.3%	3.5%	0.1%
	95	425	95	284	98	257	95	95	8705	81	84	0.0%	0.0%	0.0%
	103	763	100	393	110	469	105	92	24,650	94	81	−4.7%	11.7%	6.4%
	77	674	76	505	85	403	79	70	23,971	64	56	−3.5%	12.8%	8.8%
6	229	653	229	844	229	539	244	227	Fixed	211	202	−6.2%	7.5%	0.8%
	307	1295	307	1015	325	398	314	283	Fixed	255	275	−2.3%	11.0%	8.4%
	359	1659	391	850	427	585	372	353	Fixed	350	343	−3.5%	5.2%	1.6%
	332	1423	337	601	350	784	345	318	Fixed	328	323	−3.7%	5.2%	1.3%
	245	1581	257	845	291	401	269	234	Fixed	225	219	−8.9%	15.0%	4.7%
	247	663	257	1000	265	405	275	233	Fixed	222	207	−10.2%	17.8%	5.8%
	203	1749	217	907	217	759	216	193	Fixed	187	201	−6.0%	7.5%	1.0%
	259	890	261	1212	280	709	277	251	Fixed	236	244	−6.6%	10.4%	3.1%
	268	930	268	666	289	831	273	258	Fixed	234	248	−2.0%	5.8%	3.7%
	226	1037	215	558	215	748	238	204	Fixed	200	213	−9.8%	11.7%	0.7%
Fixed time is equal to 28,800 s											*Avg.*	−2.7%	5.1%	2.2%

(UB_{CPLEX}), is shown in the first column of the percentage deviation column. Apart from this, the deviation of the best solution obtained by CPLEX and the PSO algorithms, from the best lower bound (LB_{best}), are shown in the second and third columns of the percentage deviation column, respectively. In all runs, the best solution obtained by all PSO algorithms has been reported. As mentioned, as a result of implementing two different lower bounding methods corresponding to scenario 2 and 3 in Section 4 (LB_2 and LB_3, respectively), LB_{best} is selected from LB_{CPLEX}, LB_2, and LB_3 in order to evaluate the performance of the PSO algorithms.

The experimental results shown in Table 6 indicate the two scenarios related to the lower bounding mechanism (Section 4) are able to generate good quality lower bounds compared to LB_{CPLEX}, only when the number of machines and/or the number of jobs increase (i.e., the problem size increase) and the computational time limit decreases. In other words, during the same computational time limit, the second scenario is able to generate good quality lower bound for the original MILP model compared to LB_{CPLEX}, only when the number of violations on the size capacity restriction is not too much on the upper bound obtained by this scenario. Likewise, the third scenario is able to generate good quality lower bound for the original MILP model compared to LB_{CPLEX}, only when the number of violations on the maximum number of assigned jobs restriction is not too much on the upper bound obtained by this scenario.

The PSO algorithms are tuned based on setting parameters according to Section 5.2. Since there is exhaustive combination enumeration between batch compositions compared to the same batch composition on all machines, the solution space and, consequently, the complexity of the problem increases. Therefore, the stopping criterion of the search algorithm is considered as either Itr_{max} or 0.2 h, 0.3 h, and 0.5 h of the computational time (CT) for small-, medium-, and large-size problems, respectively.

CPLEX was able to find the optimal solution of two-machine problems related to small-size instances in around 2.5 h. For these problems, the PSO algorithms are able to find feasible solutions which are equal to the optimal solution reported by CPLEX for eight out of ten problems, but in 5.5 m on average. For other two problems, the PSO algorithms have 2.4% deviation of average from the optimal solutions reported by CPLEX. The average deviation between the best solutions reported by the PSO algorithms and CPLEX is 2.6% and 5.9% for three- and six-machine problems, respectively, in favor of the PSO algorithms.

Apart from this, Table 6 shows 5.1% and 2.2% on average deviation of the best solution obtained by CPLEX (UP_{CPLEX}) and PSO-based algorithms (UP_{PSO}), from the best lower bound (LB_{best}), respectively, with respect to all test problems. These deviations increase as the size of problems is increased from two-machine problems to six-machine problems (i.e., 9.7% and 3.1% on average deviation of UP_{CPLEX} and UP_{PSO} from LB_{best}, respectively, for six-machine problems). This indicates the efficiency of the PSO algorithms to obtain good quality solutions in a shorter computational time. It is worth noting that the result of this comparison is aligned with the result of the comparisons related to the test problems developed in Section 5.3.

6. Discussion

To further illustrate the benefit of integrating different job assignment to batches on machines into the traditional FBPP, consider a two-machine FBPP instance including seven jobs. The processing time of jobs on each machine as well as the size of jobs are provided in Table 7. Assume that the maximum number of jobs assigned to batches are 3 and 2 on the first and the second machines, respectively. The size capacity of batches on the first and the second machines are 20 and 10, respectively. With respect to the same and different batch compositions on two machines, the optimal schedules are depicted in Figs. 1 and 2, respectively.

The problem is solved optimally with minimization of makespan by restricting the MILP model to maintain the same batch composition on

Table 7
The processing time and the size of jobs.

Job	Processing time		Job size
	M_1	M_2	
j_1	13	17	6
j_2	7	13	10
j_3	13	1	4
j_4	15	13	8
j_5	17	9	1
j_6	20	7	4
j_7	18	11	10

machines. The Gantt chart of the optimal solution is presented in Fig. 1. If the assumption of different batch compositions is applied to this problem instance, we can suggest a schedule as presented in Fig. 2 with makespan equal to 67 which is 11.84% better than the solution with the same batch composition (67 compared to 76). When the same batch composition is considered for two machines, the minimum size capacity on all machines, i.e., 10, as well as the minimum value between the maximum number of jobs assigned to batches on all machines, i.e., 2, must be considered as the size capacity and the maximum number of jobs assigned to batches for each developed batch on each machine. Therefore, we have low utilization on some machines (under machine capacity). Thus, the benefits of integrating different batch compositions into the FBPP are reduction in machine idle times, i.e., higher utilization of machines, and jobs' completion time, i.e., utilizing timely processed batches on machines. It is worth noting that the optimal solution obtained by the MILP model with different batch compositions on machines is always either equal or better than the optimal solution obtained by the MILP model when the same batch composition is followed on machines in a flowshop environment.

7. Conclusions

In this research, we approached the flowshop batch processing problem with minimization of makespan as the criterion in which the batch capacity restricts both the number and the total attribute size of jobs assigned to each batch on each machine. Unlike the extensive amount of research reported on the flowshop batch processing problem, no attention has been given to investigating batch processing problem with different combinations and assignments of jobs to batches on machines, i.e., different batch compositions on machines. We proposed an MILP model to present the research problems and to optimally solve small-size problems. We showed the benefit of integrating different batch compositions on machines into the traditional FBPP. As a result, not only the completion time of jobs is reduced by utilizing timely processed batches on machines, but also idle machine times are reduced by higher utilization of machines. Since the proposed research problem is strongly NP-hard, several meta-heuristic algorithms based on PSO are proposed to solve industry-size problems. The PSO algorithms enhanced with a local search structure are different based on local search structures. These algorithms consist of a mechanism to generate the initial population, neighborhood mechanisms, and diversifying technique. Two lower bounding mechanisms are proposed for the special cases of the proposed research problem in order to generate good quality lower bounds.

The test problems are generated at three levels corresponding to the number of jobs with regard to three levels of the number of machines for each. With respect to 30 test problems at three categories, the results of the proposed PSO-based algorithms are compared to the optimal solutions/lower bounds obtained from CPLEX. Due to the complexity of the problem, CPLEX could find the optimal solution just for two-machine problems in around 2.5 h on average, while the proposed PSO-based algorithms found optimal/near optimal solutions, but in around 5.5 m. Apart from this, for the rest of the three- and six-machine problems, the proposed PSO-based algorithms found good quality solutions with respect to the best lower bound with an average percentage deviation of around 3.0%, compared to 7.7% for CPLEX. This argues strongly in favor of the efficacy and efficiency of the proposed PSO algorithms. Finally, paired *t*-test on the difference between each pair of proposed PSO algorithms in each category of machine size show the PSO algorithm enhanced with best neighbor local search has the best performance compared to the PSO algorithms with best fit and first fit local searches.

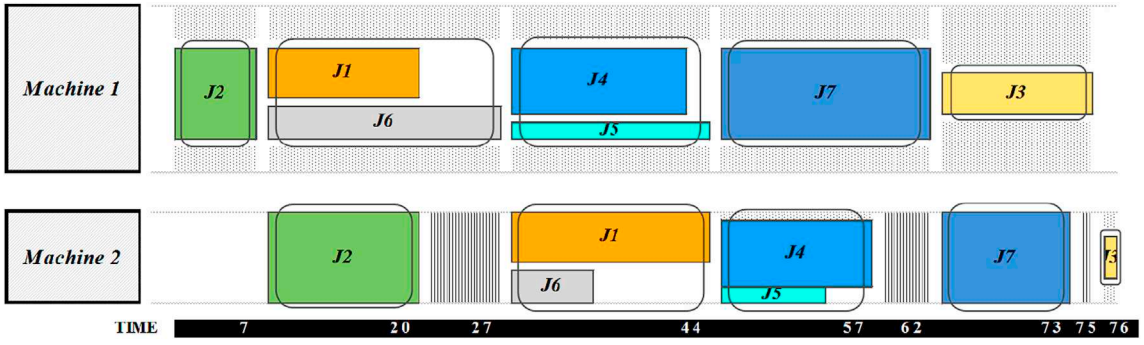


Fig. 1. Illustration of optimal batch schedules with the same batch compositions.

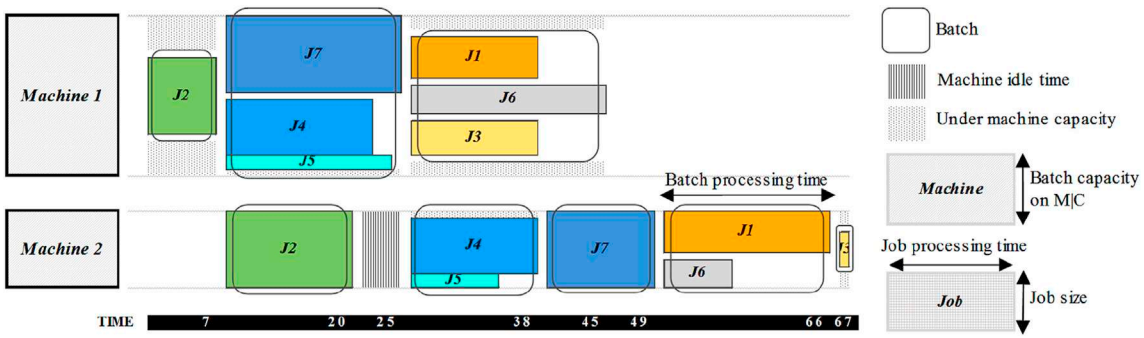


Fig. 2. Illustration of optimal batch schedules with different batch compositions.

The development and empirical evaluations in this research also show several applicable areas for future work. First, developing a tight lower bound mechanism to generate high quality lower bounds and, consequently, evaluating more accurately the performance of proposed algorithms form further investigations for future research to be pursued by the authors. Second, the performance of solutions obtained by hybrid PSO algorithms can be improved by generating a dedicated neighborhood mechanism which focuses on only effective neighbor solutions. In addition, studying other objective functions along with other shop environments as well as developing other types of hybrid algorithms remains the area to be researched in the future.

Appendix A

The PSO algorithm parameters

Swarm Size	Lower bounds	Upper bounds
w	0.4	2.0
$C1$	0.4	2.0
$C2$	0.4	2.0
V	−4.0	4.0
X	0.0	4.0
	Value	
a	10	
Itr_{max}	10,000	

Appendix B

ANOVA table on the natural log of data

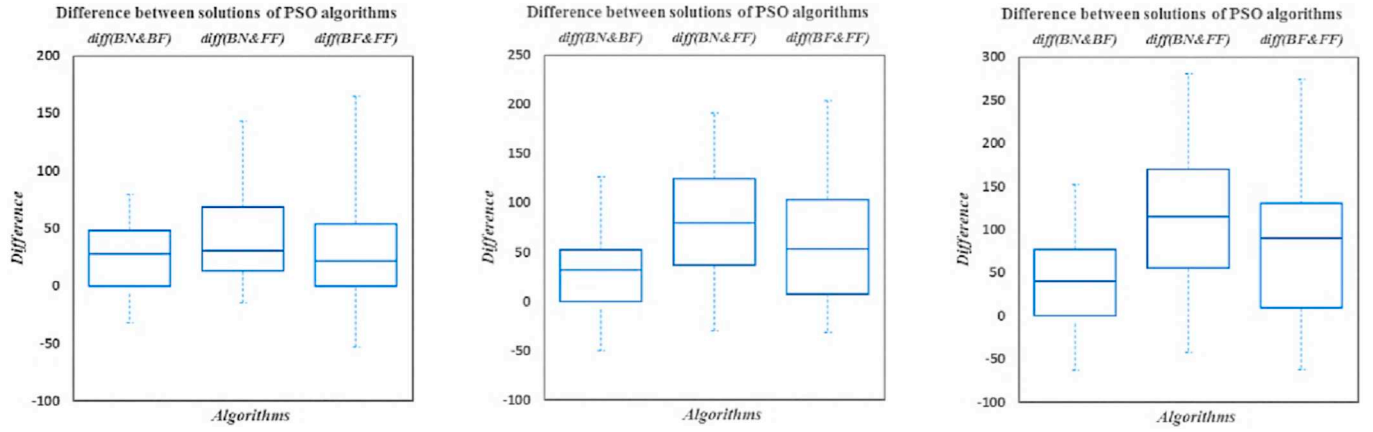
	Sum of Square	Degree of Freedom	Mean of Square	F-Statistic	P-value
Source (main plot)					
Rep (or Blocks)	16.16134	4	4.04034	0.43960	7.79914E-01
PS	4868.07704	2	2434.03852	264.83261	3.50070E-60
PT	432.56422	1	432.56422	47.06463	6.39978E-11
JS	160.40781	2	80.20391	8.72649	2.22939E-04
MN	91.93366	2	45.96683	5.00136	7.48569E-03
PS:PT	3.24093	2	1.62047	0.17631	8.38470E-01
PS:JS	21.24674	4	5.31168	0.57793	6.78923E-01
PT:JS	2.53065	2	1.26532	0.13767	8.71457E-01
PS:MN	8.73946	4	2.18486	0.23772	9.16828E-01
PT:MN	10.07743	2	5.03872	0.54823	5.78731E-01
JS:MN	40.57661	4	10.14415	1.10372	3.55571E-01
PS:PT:JS	30.95014	4	7.73754	0.84187	4.99877E-01
PS:PT:MN	30.25581	4	7.56395	0.82299	5.11679E-01
PS:JS:MN	15.84196	8	1.98025	0.21546	9.87979E-01
PT:JS:MN	20.17416	8	2.52177	0.27438	9.73770E-01
PS:PT:JS:MN	37.39380	8	4.67423	0.50857	8.49328E-01
Main Plot Error	2095.51532	228	9.19086		
Source (Sub-plot)					
Alg	3.52888	2	1.76444	53.12296	2.35857E-21
PS:Alg	0.11927	4	0.02982	0.89772	4.65174E-01
PT:Alg	0.23363	2	0.11681	3.51697	3.05424E-02
JS:Alg	0.00000	4	0.00000	0.00000	1.00000E+00
MN:Alg	0.69078	4	0.17270	5.19943	4.24082E-04
PS:PT:Alg	0.18232	4	0.04558	1.37226	2.42639E-01
PS:JS:Alg	0.08632	8	0.01079	0.32484	9.56459E-01
PT:JS:Alg	0.11465	4	0.02866	0.86299	4.86106E-01
PS:MN:Alg	0.10461	8	0.01308	0.39371	9.23920E-01
PT:MN:Alg	0.00000	4	0.00000	0.00000	1.00000E+00
JS:MN:Alg	0.11836	8	0.01479	0.44543	8.93416E-01
PS:PT:JS:Alg	0.22911	8	0.02864	0.86225	5.48452E-01
PS:PT:MN:Alg	0.24320	8	0.03040	0.91528	5.03468E-01
PS:JS:MN:Alg	0.17481	16	0.01093	0.32894	9.93948E-01
PT:JS:MN:Alg	0.21157	8	0.02645	0.79621	6.06197E-01
PS:PT:JS:MN:Alg	0.45450	16	0.02841	0.85524	6.21900E-01
Subplot Error	14.34858	432	0.03321		
Total	7906.52768	829			

Rep = Replicate; PS=Problem Size; PT=Processing time; JS = Job size; MN = Machine number; Alg = Algorithm.

Appendix C

Box plots

Two-machine problems	Three-machine problems	Six-machine problems



References

Arabameri, S., Salmasi, N., 2013. Minimization of weighted earliness and tardiness for No-wait sequence-dependent setup times flowshop scheduling problem. Comput. Ind. Eng. 64, 902–916.

Baker, K.R., Trietsch, D., 2009. Principals of Sequencing and Scheduling. Wiley.

Brucker, P., 2007. Scheduling Algorithms, fifth ed. Springer.

Campbell, H.G., Dudek, R.A., Smith, M.L., 1970. A heuristic algorithm for the n-job and m-machine sequencing problem. Manag. Sci. 16 (10), 630–637.

Damodaran, P., Srihari, K., 2004. Mixed integer formulation to minimize makespan in a flow shop with batch processing machines. Math. Comput. Model. 40, 1465–1472.

Damodaran, P., Manjeshwar, P.K., Srihari, K., 2006. Minimizing makespan on a batch-processing machine with non-identical job sizes using genetic algorithms. Int. J. Prod. Econ. 103, 882–891.

Damodaran, P., Srihari, K., Lam, S.S., 2007. Scheduling a capacitated batch-processing machine to minimize makespan. Robot. Comput. Integr. Manuf. 23, 208–216.

Hajinejad, D., Salmasi, N., Mokhtari, R., 2011. A fast hybrid particle swarm optimization algorithm for flow shop sequence dependent group scheduling problem. Sci. Iran. E 18 (3), 759–764.

IBM, 2009. ILOG CPLEX Optimization Studio, Version 12.2. .

Kennedy, J., Eberhart, R.C., 1995. Particle swarm optimization. In: Proceeding IEEE International Conference Neural Network Piscataway IEEE 4, pp. 1942–1948.

Liao, L.M., Huang, C.J., 2011. Tabu search heuristic for two-machine flowshop with batch processing machines. Comput. Ind. Eng. 60, 426–432.

Liao, Ch, Liao, L., 2008. Improved MILP models for two-machine flowshop with batch processing machine. Math. Comput. Model. 48, 1254–1264.

Liao, C.J., Tseng, C.T., Luarn, P., 2007. A discrete version of particle swarm optimization for flowshop scheduling problems. Comput. Oper. Res. 34, 3099–3111.

Manjeshwar, P.K., Damodaran, P., Srihari, K., 2009. Minimizing makespan in a flow shop with two batch-processing machine using simulated annealing. Robot. Comput. Integr. Manuf. 25 (3), 667–679.

Montgomery, D., 2009. Chapter 14. In Design and Analysis of Experiments, 7 ed. Wiley, New York.

Nawaz, M., Enscore, E., Ham, I., 1983. A heuristic algorithm for the *m*-machine, *n*-job flow-shop sequencing problem, omega. Int. J. Manag. Sci. 11 (1), 91–95.

Pinedo, M., 2012. Scheduling, Theory, Algorithms, and Systems, fourth ed. Springer.

Poli, R., Kennedy, J., Blackwell, T., 2007. Particle swarm optimization an overview. Swarm Intell. 1, 33–57.

Potts, Ch.N., Kovalyov, M.Y., 2000. Scheduling with batching: a review. Eur. J. Oper. Res. 120, 228–249.

RStudio, 2011. Integrated Development Environment for R [Computer Software]. Boston, MA.

Shahvari, O., Logendran, R., 2016. Hybrid flow shop batching and scheduling with a bi-criteria objective. Int. J. Prod. Econ. 179, 239–258.

Shahvari, O., Logendran, R., 2017. An enhanced tabu search algorithm to minimize a bi-criteria objective in batching and scheduling problems on unrelated-parallel machines with desired lower bounds on batch sizes. Comput. Oper. Res. 77, 154–176.

Shen, L., Gupta, J., Buscher, U., 2014. Flow shop batching and scheduling with sequence-dependent setup time. J. Sched. 17, 353–370.

Sung, C.S., Yoon, S.H., 1997. Minimizing maximum completion time in a two-batch processing machine flowshop with dynamic arrivals allowed. Eng. Optim. 28, 231–243.

Sung, C.S., Kim, Y.H., Yoon, S.H., 2000. A problem reduction and decomposition approach for scheduling for a flowshop of batch processing machines. Eur. J. Oper. Res. 121, 179–192.

Tadayon, B., Salmasi, N., 2012. A two criteria objective function flexible flowshop scheduling problem with machine eligibility constraint. Int. J. Adv. Manuf. Technol. 64, 1001–1015.

Tang, L., Liu, P., 2009. Minimizing makespan in a two-machine flowshop scheduling with batching and release time. Math. Comput. Model. 49, 1071–1077.

Tseng, C.T., Liao, C.J., 2008. A particle swarm optimization algorithm for hybrid flowshop scheduling with multiprocessor tasks. Int. J. Prod. Res. 46 (17), 4655–4670.

Uzsoy, R., 1994. Scheduling a single batch processing machine with non-identical job sizes. Int. J. Prod. Res. 32, 1615–1635.

Xu, R., Chen, H., Li, X., 2012. Makespan minimization on single batch-processing machine via ant colony optimization. Comput. Oper. Res. 39 (3), 582–593.