Computers & Industrial Engineering 113 (2017) 425–436

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Scheduling non-identical parallel batch processing machines to minimize total weighted tardiness using particle swarm optimization

Maria Hulett[a,*], Purushothaman Damodaran[b], Mahbod Amouie[c]

[a] Department of Management Science, University of Miami, Miami, FL 33124, USA
[b] Department of Industrial and Systems Engineering, Northern Illinois University, DeKalb, IL 60115, USA
[c] Norfolk Southern Corporation, Norfolk, VA 23510, USA

ARTICLE INFO

Keywords:
Batch processing machines
Non-identical parallel machines
Total weighted tardiness
Particle Swarm Optimization

ABSTRACT

This research aims at scheduling a set of Batch Processing Machines (BPMs) used to test printed circuit boards in an electronics manufacturing facility. The facility assembles and tests printed circuit boards (or jobs) of different sizes. The BPMs can process a batch of jobs as long as the total size of all the jobs in a batch does not exceed the machine's capacity. The objective is to minimize the total weighted tardiness, thereby minimize the total penalty incurred by the company for late deliveries. The problem under study is known to be NP-hard. Consequently, a Particle Swarm Optimization (PSO) algorithm has been proposed. Likewise, a heuristic is proposed to simultaneously group the jobs into batches and schedule them on a machine. The effectiveness of the PSO algorithm is examined using random instances and the results were compared to a differential evolution algorithm and a commercial solver used to solve a mixed-integer linear program. Experimental results indicate that the PSO algorithm is very competitive on smaller problem instances and reports better quality solutions in a short time on larger problem instances.

1. Introduction

This research work was motivated by a practical application observed at an electronics manufacturing and testing facility. The electronics manufacturer assembles and tests printed circuit boards for different customers. Depending upon the application in which the printed circuit boards are used, they vary in size. After assembling all the components to the board, the board itself is subjected to a thermal cycling test in an environmental stress screening chamber to identify any early fallouts. The chamber is capable of testing multiple printed circuit boards simultaneously as long as the boards can fit in the chamber. When the boards are grouped for testing, the testing time of the entire batch is dictated by the board that requires the longest testing time. When customers ask the manufacturer to test, they provide the minimum time for which the boards need to be tested. Consequently, the manufacturer is allowed to test a board for more than what the customer prescribed. However, if the manufacturer decides to test for a longer period of time, their machines are overused for which the customer does not pay the manufacturer. If not properly planned, poor batching and scheduling can lead to costly late deliveries. The manufacturing facility has several chambers each with different capacity. The scheduler's responsibility is to form the batches and schedule them on the chambers so that the company can avoid any penalties associated with late deliveries. The printed circuit boards are referred to as jobs and the chambers as machines, in this paper, to remain consistent with the scheduling literature.

The chambers can process several jobs simultaneously. Such machines are referred to as batch-processing machines (BPMs) in the literature. In scheduling jobs on these machines, the jobs are first grouped into batches and later the batches are scheduled on the machines. As the jobs are grouped into batches, the start time and completion time of all the jobs in a batch are identical. In the application under study, the sizes of the jobs vary and the capacity of the machines are non-identical. The BPM can process a batch of jobs as long as the total size of all the jobs does not exceed its capacity. BPMs occur frequently in manufacturing and service environments such as semiconductor burn-in operations, environmental stress screening chambers, chemical processes in tanks and kilns, wafer fabrication process, testing electrical circuits, heat-treating furnaces in the metalworking industries, material handling and transportation vehicles, food and pharmaceutical industries, among others.

The objective of the problem under study is to minimize total weighted tardiness (i.e. total weighted tardy deliveries from the company's perspective). The scheduling problem considered in this research

* Corresponding author.
E-mail address: mhulett@bus.miami.edu (M. Hulett).

http://dx.doi.org/10.1016/j.cie.2017.09.037
Received 24 October 2016; Received in revised form 18 September 2017; Accepted 20 September 2017
Available online 21 September 2017
0360-8352/ © 2017 Elsevier Ltd. All rights reserved.

is NP-hard, consequently a Particle Swarm Optimization (PSO) algorithm is proposed and its performance is evaluated in the paper through an experimental study.

The remainder of the paper is organized as follows. The scheduling problem under study is described in Section 2. The literature reviewed on BPM scheduling problems and PSO are presented in Section 3. A mathematical formulation of the problem under study is presented in Section 4. The proposed PSO algorithm and the experimental study conducted is explained in detail in Section 5. Section 6 presents the conclusions.

2. Problem description

Given a set J of n jobs and a set M of q non-identical BPMs, the scheduling problem under study is to determine how the jobs should be batched and scheduled on the BPMs to minimize the Total Weighted Tardiness (TWT). The BPMs can process a batch of jobs as long as the total size of all the jobs in a batch does not exceed the machine capacity in which it is processed. Since there are only n jobs, the maximum number of batches needed is also n (i.e. assuming one job per batch will result in n batches). Once the processing of a batch is started, it cannot be stopped prematurely.

The capacities (S_m) of the machines ($m \in M$) are non-identical. The processing time (p_j), size (s_j), and weight or priority (w_j) of each job ($j \in J$) are given. Each job can be processed on any BPM. Any job can be grouped with any other job as this study does not consider job incompatibility. There is also no precedence relationship among the jobs. All jobs in a batch are processed simultaneously. The processing time of batch b in machine m is P_{bm} and it is equal to the longest processing time of all the jobs that constitute the batch. Once a machine starts processing a batch, no additional jobs can be introduced or removed from the batch. Two interdependent decisions are to be made to schedule the jobs on the machines. These decisions include (1) how to group the jobs into batches and (2) how to schedule the batches formed on a machine. These decisions cannot be separated as the composition of the batch affects the batch processing time and, hence, the tardiness.

The problem under study can be denoted using the standard three field notation in scheduling literature $\alpha|\beta|\gamma$ of Graham, Lawler, Lenstra, and Kan (1979) as $Pm|p\text{-}batch, s_j, S_m|TWT$. When the sizes of all the jobs are equal and if there is only one machine, whose capacity is such that it can process only one job at a time, then the problem under study reduces to a single (or discrete) machine scheduling problem with TWT as the objective (i.e. $1||TWT$). Since this reduced problem was proven to be NP-hard by Lawler (1977), the problem under study is also NP-hard. Commercial solvers may require prohibitively long computational time to find optimal solutions for the problem under study (or in general any NP-hard scheduling problem). Consequently, heuristics and metaheuristics are commonly developed to solve difficult to solve scheduling problems. This research proposes a Particle Swarm Optimization (PSO) algorithm to solve the problem under study and evaluates its performance, in terms of solution quality and computational time, by comparing the results with a commercial solver used to solve the mathematical formulation and a differential evolution algorithm (Amouie, 2014). A brief review of successful application of PSO to solve a variety of scheduling problems can be found in the literature review section.

3. Literature review

Research work on scheduling batch processing machines has received attention from a variety of fields (including metal working industry, chemical processing, and electronics manufacturing). One of the pioneers in this field is Ikura and Gimple (1986) who presented efficient algorithms to schedule a single batch processing machine. Later, significant research efforts in the development of algorithms have been proposed. Lee, Uzsoy, and Martin-Vega (1992) presented dynamic programming based algorithms to minimize several performance measures on a single batch processing machine. They also extended a number of heuristics developed for parallel identical unit processing machines to the case of parallel batch processing machines. Chandru, Lee, and Uzsoy (1993) proposed heuristics to minimize the total completion time on identical parallel batch processing machines. Uzsoy (1995) examined a number of problems related to the scheduling of batch processing machines with incompatible job families. He developed efficient optimal algorithms to minimize makespan ($C_{\max}$), maximum lateness ($L_{\max}$) and total weighted completion time for static problems where all jobs are available simultaneously. This work was also extended to problems with parallel identical batch processing machines. Problems with dynamic job arrivals were later considered and an efficient optimal algorithm for minimizing $C_{\max}$ and several heuristics to minimize $L_{\max}$ were provided. Brucker, Mikhail, and Shafransky (1998) examined batch scheduling on parallel machines with respect to deadlines. They showed that the problem is NP-hard for two identical machines, unit processing times, unit set-up times, and a common deadline. A family of approximation algorithms for the general problem with unrelated machines was constructed and a new dynamic rounding technique was developed. The computational complexity for their algorithm was also examined. A review on scheduling with batching, giving details of the basic algorithms, and referencing other significant results is presented in Potts and Kovalyov (2000). Moreover, various heuristics have been developed to schedule batch processing machines. A list of these studies is Wang and Uzsoy (2002), Mönch, Balasubramanian, Fowler, and Pfund (2003), Balasubramanian, Mönch, Fowler, and Pfund (2004), Chang, Damodaran, and Melouk (2004), Damodaran, Manjeshwar, and Srihari (2006), Malve and Uszoy (2007), Kashan and Karimi (2008), Raghavan and Venkataramana (2009), Damodaran, Velez-Gallego, and Maya (2009), Damodaran and Vélez-Gallego (2012), Damodaran, Ghrayeb, and Guttikonda (2013) and Liu, Ng, and Cheng (2014) among others. On-line algorithms for scheduling on parallel batch processing machines have been proposed in Nong, Cheng, and Ng (2008) and Yuan, Ng, and Cheng (2011) as well.

There are several studies that address tardiness based scheduling objectives on parallel batch processing machines. Kurz (2003) discussed results on the structure of job-to-batch assignments in an optimal schedule for minimizing total weighted tardiness in the parallel batch processing machine scheduling problem. Mönch et al. (2003) attempted to minimize total weighted tardiness on parallel batch machines with incompatible job families and unequal ready times of the jobs. They proposed two different decomposition approaches. The first approach forms fixed batches, then assigns these batches to the machines using a genetic algorithm (GA), and finally sequences the batches on individual machines. The second approach first assigns jobs to machines using a GA, then forms batches on each machine for the jobs assigned to it, and finally sequences these batches. Raghavan and Venkataramana (2009) developed an efficient algorithm for solving the scheduling problem in a system of parallel processors with the objective of minimizing total weighted tardiness. They proposed the ant colony optimization approach for this problem. Liu, Ng, and Cheng (2009) considered the problem of scheduling jobs with release dates on parallel unbounded batch processing machines to minimize the maximum lateness. They developed a polynomial-time approximation scheme for the problem to minimize the maximum delivery completion time, which is equivalent to minimizing the maximum lateness from the optimization viewpoint. Chiang, Cheng, and Fu (2010) proposed a memetic algorithm with a new genome encoding scheme to search for the optimal or near-optimal batch formation and batch sequence simultaneously. They addressed the problem of scheduling identical parallel batch machines with incompatible job families and dynamic job arrival. More recently, Gokhale and Mathirajan (2014) considered unequal release times, incompatible job families, non-identical job sizes, heterogeneous batch processors, and allowance for job splitting in the system of parallel processing with the objective of minimizing total weighted tardiness. They proposed heuristic algorithms to address this problem. Amouie

(2014) considered the same problem under study in this research and proposed a heuristic based on column generation and a Differential Evolution (DE) meta-heuristic to minimize the weighted tardiness for non-identical parallel batch processing machines.

PSO is a metaheuristic commonly used to solve machine scheduling problems in the literature. Based on the metaphor of social interaction and communication such as bird flocking and fish schooling, PSO was first introduced to optimize various continuous nonlinear functions by Kennedy and Eberhart (1995). Later, the algorithm was modified to introduce a new parameter called inertia weight (Shi & Eberhart, 1998) and reworked to operate on discrete binary variables (Kennedy & Eberhart, 1997). PSO is particularly different from other evolutionary-type methods because it does not use the filtering operation (such as crossover and/or mutation), and the members of the entire population are maintained throughout the search procedure so that information is socially shared among individuals to direct the search towards the best position in the search space (Tasgetiren, Liang, Sevkli, & Gencyilmaz, 2007). This algorithm has been successfully applied in several scheduling problem as the singe machine scheduling (Anghinolfi & Paolucci, 2009), parallel machine scheduling (Kashan & Karimi, 2009; Lin, 2013; Niu, Zhou, & Wang, 2010), flow shop scheduling (Liao, Tseng, & Luarn, 2007; Pan, Tasgetiren, & Liang, 2008; Tang & Wang, 2013; Tasgetiren et al., 2007; Yang, Chou, & Chang, 2014), job shop scheduling (Shao, Weiqi, Liu, & Zhang, 2013), and patient scheduling (Kanagaa & Valarmathi, 2012), among others. However, in the literature, studies that have approached the parallel batch processing machine scheduling problem using PSO algorithms are limited. Damodaran, Diyadawagamage, Ghrayeb, and Vélez-Gallego (2012) proposed a PSO algorithm for minimizing makespan of non-identical parallel batch processing machines. Chang, Chen, and Ma (2013) studied the scheduling problem of minimizing maximum lateness on parallel identical batch processing machines with dynamic job arrivals and incompatible job families. Damodaran, Rao, and Mestry (2013) addressed the multi-stage flow shop scheduling problem. They proposed a PSO algorithm, with three different heuristics to update the particle's positions, for scheduling batch processing machines such that the makespan is minimized. To the best of our knowledge, there is no published literature on the problem under study with the characteristics considered in this research using the PSO algorithm. The effectiveness of the proposed PSO algorithm is evaluated in this research by comparing the solutions from the proposed PSO approach with a DE algorithm proposed in Amouie (2014). The solution quality is also evaluated by comparing the PSO solution with a commercial solver used to solve the formulation presented in Section 4.

4. Mathematical formulation

The problem under study can be formulated as a mixed integer linear program. The notation used in the formulation is presented below.

Sets	
$\{j \in J\}$	Set of jobs
$\{m \in M\}$	Set of machines
$\{b \in B\}$	Set of batches on each machine
Parameters	
p_j	Processing time of job j
s_j	Size of job j
d_j	Due date of job j
w_j	Weight (priority) of job j
S_m	Capacity of machine m
$BigM$	A very large number
Decision variables	
P_{bm}	Processing time of bth batch scheduled on machine m
C_{bm}	Completion time of bth batch scheduled on machine m
c_j	Completion time of job j
T_j	Tardiness of job j
X_{jbm}	$\begin{cases}1, & \text{if job } j \text{ is assigned to the } b\text{th batch processed on machine } m\\ 0, & \text{otherwise}\end{cases}$

The mixed-integer linear formulation for the problem under study is presented below.

$$\text{Minimize} \quad \sum_{j\in J} w_j T_j \tag{1}$$

Subject to

$$\sum_{b\in B}\sum_{m\in M} X_{jbm} = 1 \quad j \in J \tag{2}$$

$$\sum_{j\in J} s_j X_{jbm} \leqslant s_m \quad \forall\ b \in B, m \in M \tag{3}$$

$$P_{bm} \geqslant p_j X_{jbm} \quad \forall\ j \in J,\ b \in B,\ m \in M \tag{4}$$

$$C_{1m} = P_{1m} \quad \forall\ m \in M \tag{5}$$

$$C_{bm} = C_{b-1,m} + P_{bm} \quad \forall\ b \in B/\{1\},\ m \in M \tag{6}$$

$$c_j \geqslant C_{bm} + BigM\,(1-X_{jbm}) \quad \forall\ b \in B,\ m \in M \tag{7}$$

$$T_j \geqslant c_j - d_j \quad j \in J \tag{8}$$

$$P_{bm} \geqslant 0 \quad \forall\ b \in B,\ m \in M \tag{9}$$

$$C_{bm} \geqslant 0 \quad \forall\ b \in B,\ m \in M \tag{10}$$

$$C_j \geqslant 0 \quad j \in J \tag{11}$$

$$X_{jbm} \in \{0,1\} \quad \forall\ j \in J,\ b \in B,\ m \in M \tag{12}$$

$$T_j \geqslant 0 \quad j \in J \tag{13}$$

The objective function (1) aims to minimize the total weighted tardiness. Constraint set (2) ensures that each job is processed exactly in one batch on one machine. Constraint set (3) ensures that the machine capacity is not violated. It ensures that for each batch formed, the total size of all the jobs in that batch does not exceed the capacity of the machine in which it is scheduled for processing. Constraint set (4) ensures that the processing time of the bth batch on machine m is at least equal to the longest processing time job in the batch. Constraint set (5) determines the completion time of the first batch on each machine. Constraint set (6) calculates the completion time of the bth batch on machine m. Constraint set (7) ensures that the completion time of job j is not less than the completion time of the batch in which it is processed. Constraint set (8) helps to determine the tardiness of job j based on its completion time and due date. The tardiness of a job cannot be less than the difference between completion time of the job and its due date. Constraint sets (9)–(13) impose the non-negativity and binary restrictions on the decision variables.

Even in the most straightforward scheduling environment, TWT is a difficult objective function to minimize (Lawler, 1977). Our experimental study (see Section 6) shows that IBM ILOG CPLEX, a commercial solver widely used to solve mixed integer linear programs, takes prohibitively long CPU (or run) time to solve many of the problem instances generated for the problem under study to optimality. Therefore, a PSO algorithm is proposed in this research.

5. Proposed particle swarm optimization algorithm

PSO is one of the evolutionary optimization techniques. Unlike other evolutionary-type methods, PSO maintain the members of the entire population throughout the search procedure (Kennedy, Eberhart, & Shi, 2001). PSO executes a population-based search procedure in which the exploring agents, called *particles*, move around in a multi-dimensional search space. The movement of the particle depends on its current position and the velocity. The velocity of the particle is determined by a component moving the particle towards the best position so far achieved by the particle itself (i.e. particle's own experience), and a component moving the particle towards the best solution so far achieved in its restricted neighborhood (local neighborhood) or by any in the whole swarm (global neighborhood). That is, in each iteration, each particle tries to find a new position which will improve its fitness value, based on its previous experience and also on the position of a particle which has the best known fitness value (Damodaran et al., 2012).

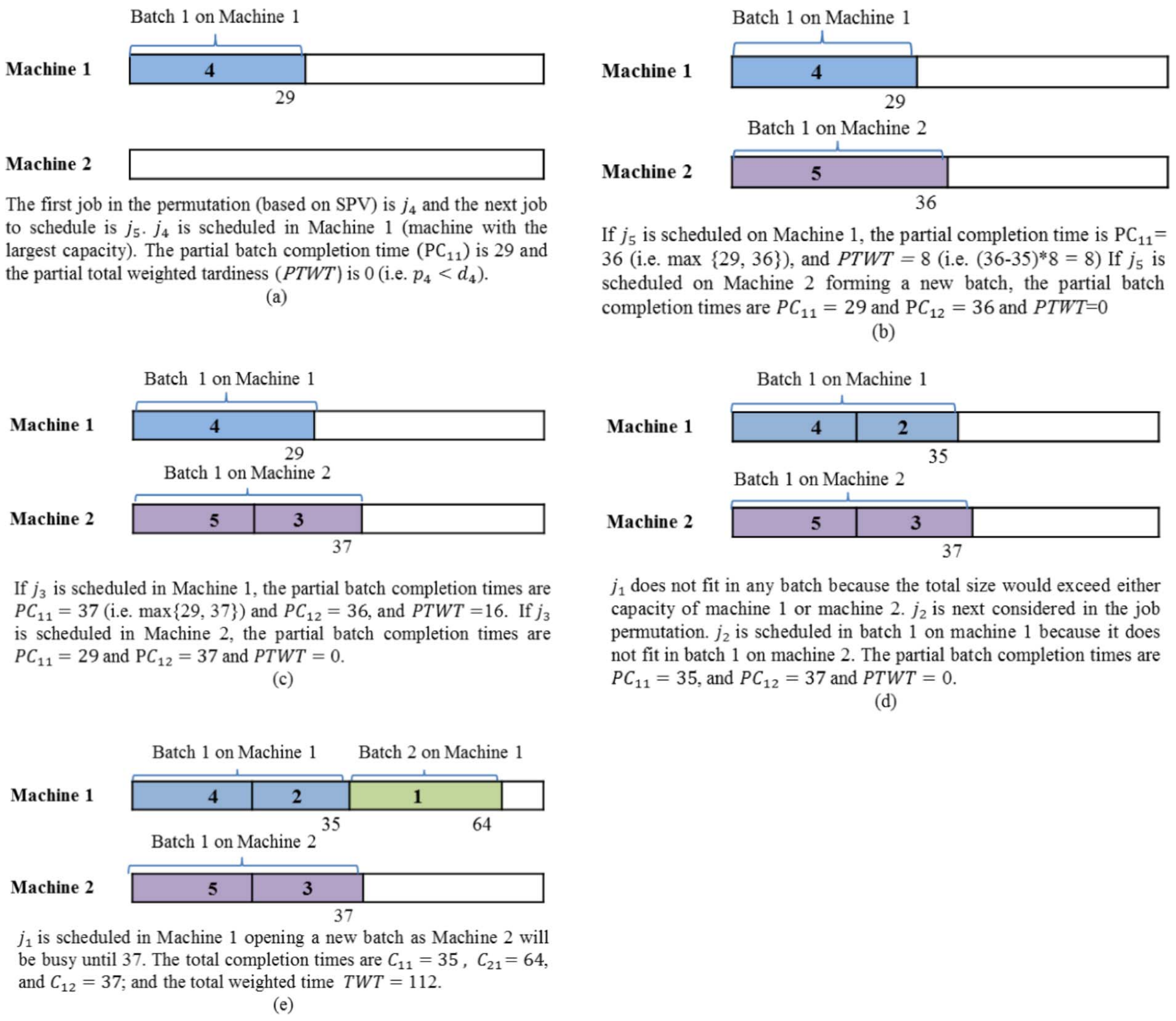
5.1. Solution representation

Solution representation is very important in designing the PSO algorithm. For the problem under study, each dimension represents a job ($j = 1,\dots,n$). The position of the ith particle (i.e. potential solution) with respect to the jth job in tth iteration is represented as $X_i^t = [X_{i1}^t,\dots,X_{ij}^t,\dots,X_{in}^t]$ (i.e. position information for n number of jobs). At each iteration t, a set of p particles is maintained in the population (i.e. $n_{pop}^t = [X_1^t,\dots,X_p^t]$). Each particle is assigned a continuous set of position values. The smallest position value (SPV) rule is then used to convert the continuous position values of the particle to a discrete job permutation (Tasgetiren, Liang, Sevkli, & Gencyilmaz, 2004). The permutation sequence (based on SPV value) is described as $S_i^t = [s_{i1}^t,\dots,s_{ij}^t,\dots,s_{in}^t]$, where s_{ij}^t is the assignment of job j of the particle i in the processing order at iteration t. According to the SPV rule, the jobs with smaller position values will be considered first for batching. A simple five job instance is shown in Table 1 to illustrate the SPV rule. The smallest position value is $X_{i4}^t = 0.8$, so job 4 is assigned to the first position of the job permutation according to the SPV rule. Job 5 is assigned to the second position of the job permutation because of its second smallest position value. In the same way, other jobs are assigned to their corresponding position of the job permutation according to their position values.

Table 1
Solution representation of particle X_{ij}^t for a five-job instance.

	Jobs				
	1	2	3	4	5
p_j	29	35	37	29	36
s_j	14	6	19	39	20
d_j	50	36	39	35	38
w_j	8	2	5	8	1
X_{ij}^t	2	4	1.33	0.8	1
s_{ij}^t	4	5	3	1	2

Using the job permutation, obtained by applying the SPV rule, the jobs are grouped and scheduled on the machines. A heuristic is



The first job in the permutation (based on SPV) is j_4 and the next job to schedule is j_5. j_4 is scheduled in Machine 1 (machine with the largest capacity). The partial batch completion time (PC_{11}) is 29 and the partial total weighted tardiness ($PTWT$) is 0 (i.e. $p_4 < d_4$).

If j_5 is scheduled on Machine 1, the partial completion time is PC_{11}= 36 (i.e. max {29, 36}), and $PTWT$ = 8 (i.e. (36-35)*8 = 8) If j_5 is scheduled on Machine 2 forming a new batch, the partial batch completion times are $PC_{11} = 29$ and $PC_{12} = 36$ and $PTWT$=0

If j_3 is scheduled in Machine 1, the partial batch completion times are $PC_{11} = 37$ (i.e. max{29, 37}) and $PC_{12} = 36$, and $PTWT$ =16. If j_3 is scheduled in Machine 2, the partial batch completion times are $PC_{11} = 29$ and $PC_{12} = 37$ and $PTWT = 0$.

j_1 does not fit in any batch because the total size would exceed either capacity of machine 1 or machine 2. j_2 is next considered in the job permutation. j_2 is scheduled in batch 1 on machine 1 because it does not fit in batch 1 on machine 2. The partial batch completion times are $PC_{11} = 35$, and $PC_{12} = 37$ and $PTWT = 0$.

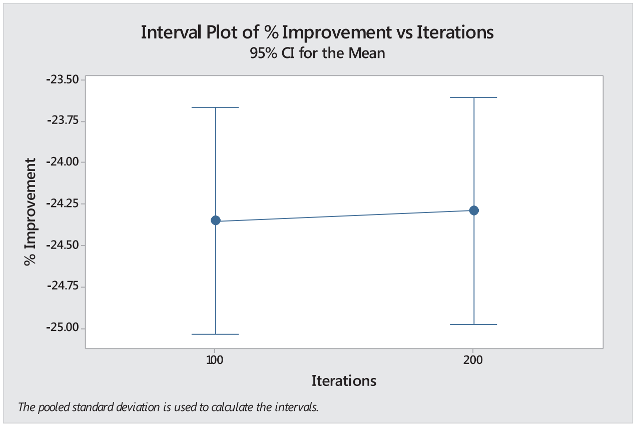
j_1 is scheduled in Machine 1 opening a new batch as Machine 2 will be busy until 37. The total completion times are $C_{11} = 35$, $C_{21} = 64$, and $C_{12} = 37$; and the total weighted time $TWT = 112$.

Fig. 1. An example to illustrate the batch forming heuristic.

proposed to simultaneously group the jobs into batches and schedule them on a machine. In this heuristic, whenever a machine is free, the unscheduled jobs are chosen based on the position values to form the batch. Initially, all the machines are free, and the machine with the largest capacity is chosen. One job at a time is considered from the job permutation to form a batch such that the machine capacity is not violated. If a job does not fit in any batch, the next job in the job permutation is considered. As soon as a batch is full or if there are no more jobs which can fit in the batch, then the batch is closed and a new batch is opened on the machine. When a job can fit in more than one existing open batch, then the job is assigned to a batch which results in minimizing the partial total weighted tardiness. Fig. 1 illustrates the heuristic proposed to form batches and how these batches are scheduled. The data shown in Table 1 is used to schedule two non-identical BPMs with capacities 50 and 40.

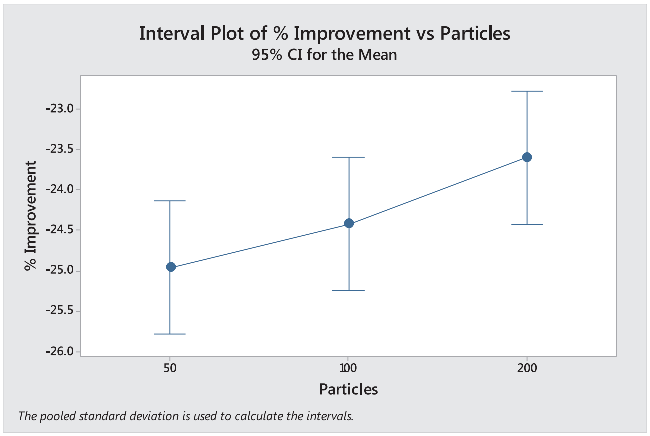
5.2. Initial population

In PSO, a population of particles (n_{pop}) are maintained through all the iterations. The first seven particles of the population are generated by first developing a job permutation using simple rules such as the earliest due-date (EDD) rule, the earliest weighted due-date (EWDD) rule, the shortest processing time (SPT) rule, weighted shortest processing time (WSPT) rule, minimum slack time (MST) rule, largest processing time (LPT) rule, and apparent tardiness cost (ATC) rule. Once the job sequence is determined, the heuristic discussed in Section



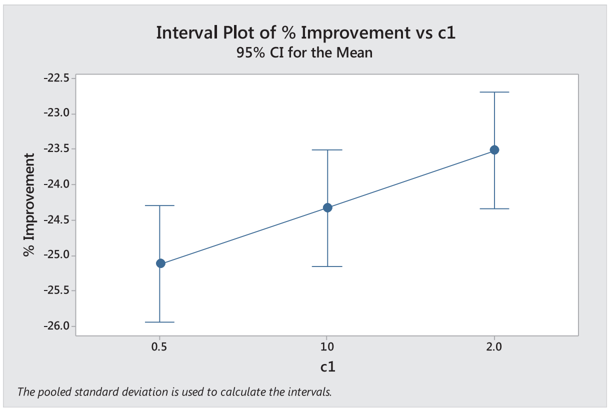
```
Analysis of Variance

Source        DF    Adj SS    Adj MS  F-Value  P-Value
Iterations     1  0.000017  0.000017     0.02    0.896
Error        160  0.155686  0.000973
Total        161  0.155702
```

Fig. 2. Statistical analysis of the number of iterations parameter for small problem instances.



```
Analysis of Variance

Source      DF    Adj SS    Adj MS  F-Value  P-Value
Particles    2  0.005057  0.002528     2.67    0.072
Error      159  0.150646  0.000947
Total      161  0.155702
```

Fig. 3. Statistical analysis of the number of particles parameter for small problem instances.



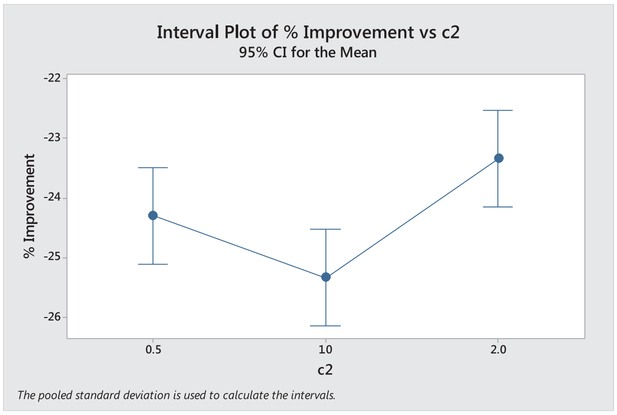
```
Analysis of Variance

Source   DF    Adj SS    Adj MS  F-Value  P-Value
c1        2  0.006924  0.003462     3.70    0.027
Error   159  0.148778  0.000936
Total   161  0.155702
```

Fig. 4. Statistical analysis of the cognitive coefficient parameter for small problem instances.

5.1 is applied to develop a schedule. The different rules applied to develop the job sequence is presented below:

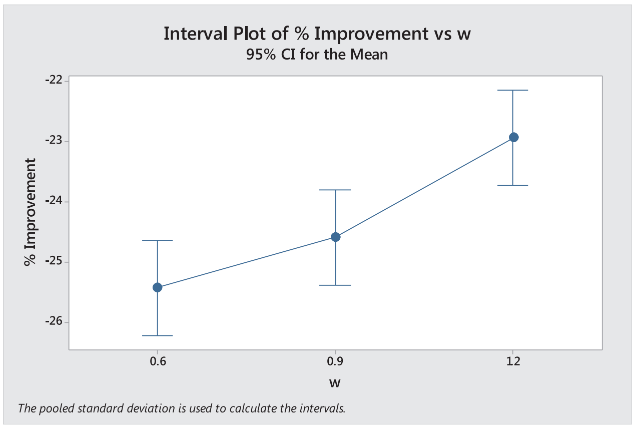
- EDD – jobs are arranged in non-decreasing order of their due dates (d_j)
- EWDD – jobs are sequenced in non-decreasing order of (d_j/w_j)
- SPT – jobs are sequenced in non-decreasing of their processing times (p_j)
- WSPT – jobs are sequenced in the decreasing order of (w_j/p_j)
- MST – jobs are sequenced in non-decreasing order of ($d_j - p_j$)
- LPT – jobs are sequenced in decreasing order of (p_j)
- ATC – jobs are sequenced in decreasing order of a ranking index $I_j(l) = \frac{w_j}{p_j} e^{\left(-\frac{\max(dj-pj-l,0)}{K\bar{p}}\right)}$, which is a function of the time l at which the machine becomes free as well as of the p_j, the w_j, and the d_j of the remaining jobs. K is the scaling parameter that can be determined empirically, and $\bar{p}$ is the average of the processing times of the remaining jobs.

Since these seven solutions are discrete job permutations, they should be converted into seven particles with continuous position values. For a given job permutation S_i^t, the position value of job is calculated by Eq. (14). The continuous position values of the other (n_{pop}−7) solutions are randomly generated according to the Eq. (15).



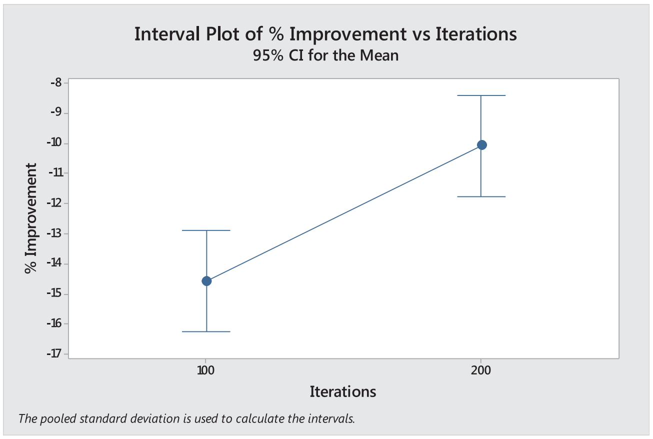
```
Analysis of Variance

Source   DF   Adj SS    Adj MS    F-Value  P-Value
c2        2  0.01072  0.005360     5.88     0.003
Error   159  0.14498  0.000912
Total   161  0.15570
```

Fig. 5. Statistical analysis of the social learning parameter for small problem instances.



```
Analysis of Variance

Source   DF   Adj SS    Adj MS    F-Value  P-Value
w         2  0.01731  0.008657     9.95     0.000
Error   159  0.13839  0.000870
Total   161  0.15570
```

Fig. 6. Statistical analysis of the inertia weight parameter for small problem instances.



```
Analysis of Variance

Source       DF   Adj SS    Adj MS    F-Value  P-Value
Iterations    1  0.08162  0.081624    14.01     0.000
Error       160  0.93242  0.005828
Total       161  1.01404
```

Fig. 7. Statistical analysis of the number of iterations parameter for large problem instances.

Typically $X_{min} = 0.0$ and $X_{max} = 4.0$ and r_1 is a uniform random number between 0 and 1.The initial velocities for the particles are also generated by a random scheme that is similar to the position value equation (see Eq. (16)), where $V_{min} = -4.0$ and $V_{max} = 4.0$ and r_2 is a uniform random number between 0 and 1.

$$X_i^t = \left[\frac{X_{max}}{n},\ldots,\frac{X_{max}}{n-j+1},\ldots,\frac{X_{max}}{1}\right] \tag{14}$$

$$X_{ij}^0 = X_{min} + (X_{max}-X_{min})*r_1 \tag{15}$$

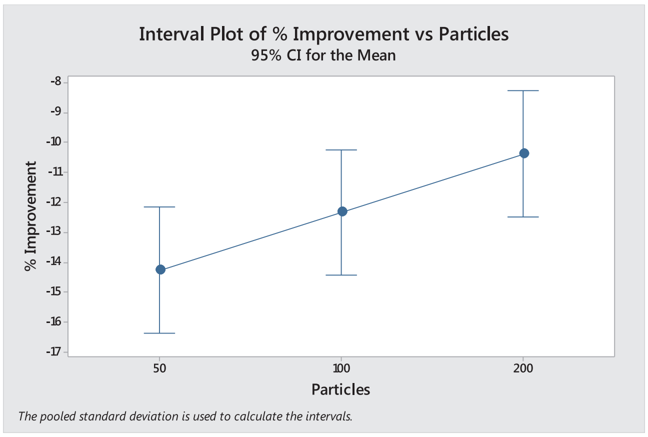
$$V_{ij}^0 = V_{min} + (V_{max}-V_{min})*r_2 \tag{16}$$

5.3. Particle updates

The behavior of a particle, in each iteration t, is a compromise among three possible alternatives: following its current pattern of exploration; going back towards its best previous position; going back towards the best historic value of all particles (Kashan & Karimi, 2009). The velocity of each particle in tth iteration is updated using Eq. (17).

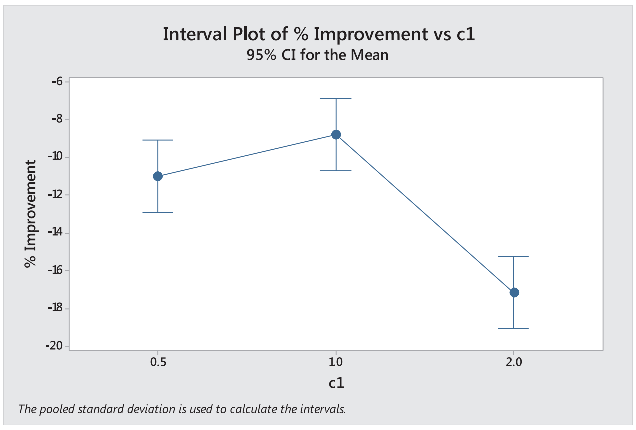
$$V_{ij}^t = w^{t-1}V_{ij}^{t-1} + c_1r_1(pbest_{ij}-X_{ij}^{t-1}) + c_2r_2(gbest_j-X_{ij}^{t-1}) \tag{17}$$

where w is the inertia weight which is a constant value chosen by the user to control the impact of the previous velocity on the current



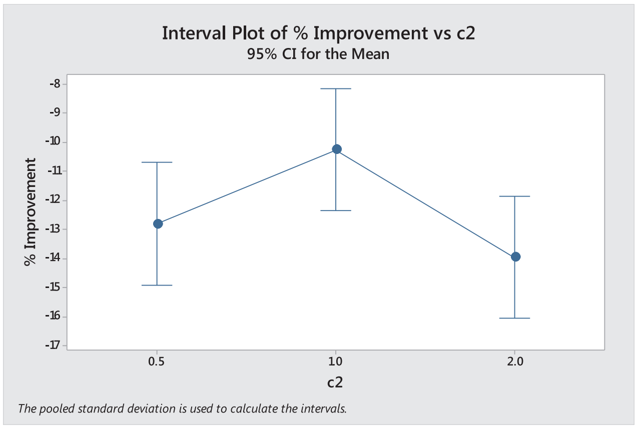
```
Analysis of Variance

Source      DF   Adj SS     Adj MS   F-Value  P-Value
Particles    2  0.04041   0.020207     3.30     0.039
Error      159  0.97362   0.006123
Total      161  1.01404
```

Fig. 8. Statistical analysis of the number of particles parameter for large problem instances.



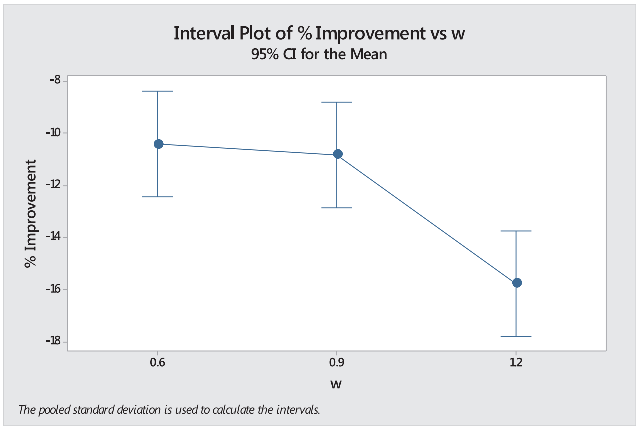
```
Analysis of Variance

Source    DF   Adj SS     Adj MS   F-Value   P-Value
c1         2  0.2017   0.100830    19.73     0.000
Error    159  0.8124   0.005109
Total    161  1.0140
```

Fig. 9. Statistical analysis of the cognitive coefficient parameter for large problem instances.



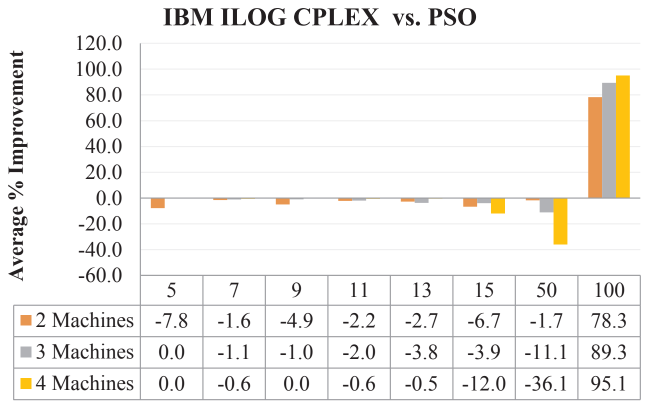
```
Analysis of Variance

Source    DF    Adj SS     Adj MS   F-Value   P-Value
c2         2  0.03872   0.019358     3.16     0.045
Error    159  0.97532   0.006134
Total    161  1.01404
```

Fig. 10. Statistical analysis of the social learning parameter for large problem instances.



```
Analysis of Variance

Source   DF    Adj SS    Adj MS  F-Value  P-Value
w         2  0.09616  0.048079     8.33    0.000
Error   159  0.91788  0.005773
Total   161  1.01404
```

Fig. 11. Statistical analysis of the inertia weight parameter for large problem instances.



	5	7	9	11	13	15	50	100
🟧 2 Machines	-7.8	-1.6	-4.9	-2.2	-2.7	-6.7	-1.7	78.3
⬜ 3 Machines	0.0	-1.1	-1.0	-2.0	-3.8	-3.9	-11.1	89.3
🟨 4 Machines	0.0	-0.6	0.0	-0.6	-0.5	-12.0	-36.1	95.1

Fig. 12. Experimental results of IBM ILOG CPLEX vs. PSO with a due date adjustment factor of 0.2.

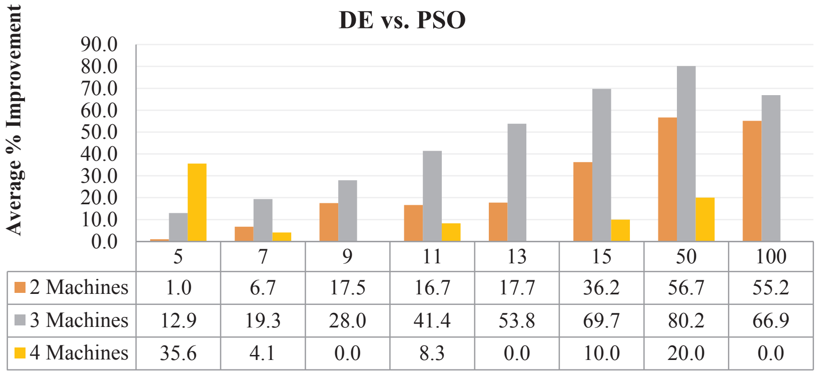
velocity. Inertia weight is updated using $w^t = \alpha.\, w^{t-1}$, where α is a decrement factor. Parameters c_1 and c_2 are the cognitive and social learning coefficients, while r_1 and r_2 are uniform random numbers between 0 and 1. $pbest_{ij}$ is the best position associated with the best fitness value for job j in particle i so far in the iterative search. It can be represented as $Pbest_i = [pbest_{i1},\dots,pbest_{ij},\dots,pbest_{in}]$. $gbest_j$ denotes the best position value for job j corresponding to the particle with the best fitness value in the entire population. It is defined as $Gbest = [gbest_1,\dots,gbest_j,\dots,gbest_n]$.

Eq. (18) is used to generate the new position, where X_{ij}^t is the new position to be generated, and X_{ij}^{t-1} is the position of the particle from the previous iteration.

$$X_{ij}^t = X_{ij}^{t-1} + V_{ij}^t \tag{18}$$

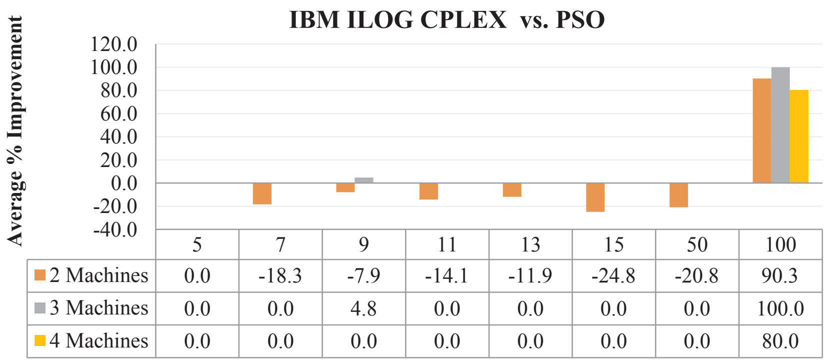
The pseudo code for the proposed algorithm can be summarized as follows:

```
BEGIN
  Initialize:
    Global best $f(Gbest) \leftarrow +\infty$
    Set iteration $t \leftarrow 0$
    Create $X_1^0$ using EDD rule, $X_2^0$ using EWDD rule, $X_3^0$ using SPT rule, $X_4^0$ using WSPT rule, $X_5^0$ using MST rule, $X_6^0$ using LPT rule, and $X_7^0$ using ATC rule
  for $i$ = 1: 7particles
    Convert the discrete job permutations into seven particles with continuous position values (using Eq. (14)). Develop a schedule using the heuristic discussed in Section 5.1 and determine the TWT.
  end for
  for $i$ = 8: $n_{pop}$
    Create the remaining initial positions $X_{ij}^0$ (using Eq. (15)) for each particle $i$. Develop a schedule using the heuristic discussed in Section 5.1 and determine the TWT.
  end for
  for $i$ = 1: $n_{pop}$
    Update personal best position $Pbest_i$
    Generate initial velocity $V_{ij}^0$ (using Eq. (16))
    if $f(Pbest_i) < f(Gbest)$ then
      Update $Gbest$
    end if
  end for
  while $t$<Fixed number of iterations or $Gbest > 0$
    do
      for all particles $i \in I$
        Compute the velocity $V_{ij}^t$ for the $t$th iteration (using Eq.
```



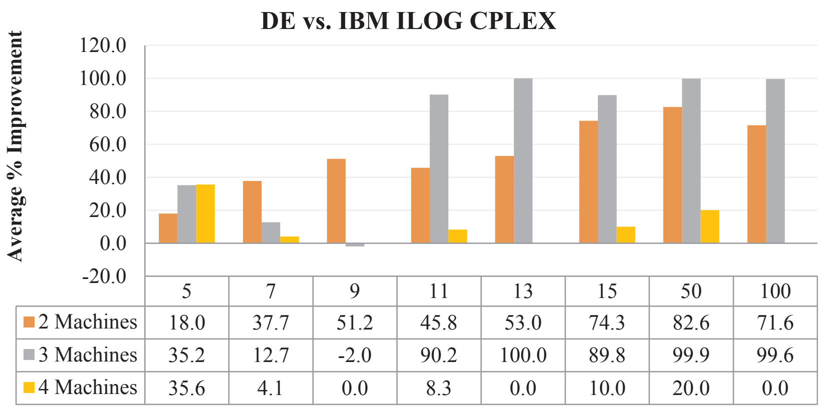
	5	7	9	11	13	15	50	100
🟧 2 Machines	1.0	6.7	17.5	16.7	17.7	36.2	56.7	55.2
⬜ 3 Machines	12.9	19.3	28.0	41.4	53.8	69.7	80.2	66.9
🟨 4 Machines	35.6	4.1	0.0	8.3	0.0	10.0	20.0	0.0

Fig. 13. Experimental results of DE vs. PSO with a due date adjustment factor of 0.2.



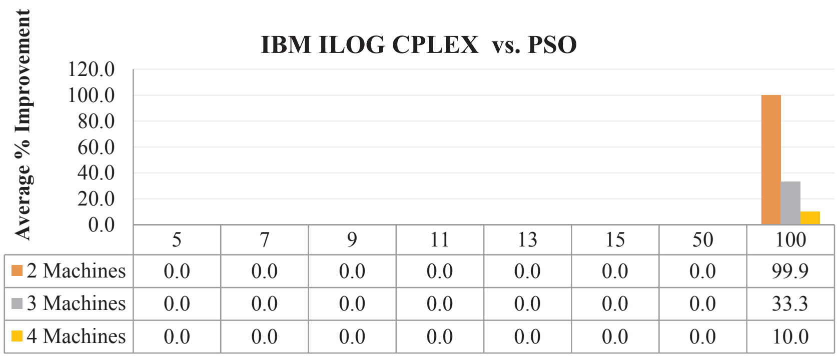
	5	7	9	11	13	15	50	100
🟧 2 Machines	0.0	-18.3	-7.9	-14.1	-11.9	-24.8	-20.8	90.3
⬜ 3 Machines	0.0	0.0	4.8	0.0	0.0	0.0	0.0	100.0
🟨 4 Machines	0.0	0.0	0.0	0.0	0.0	0.0	0.0	80.0

Fig. 14. Experimental results of IBM ILOG CPLEX vs. PSO with a due date adjustment factor of 0.33.



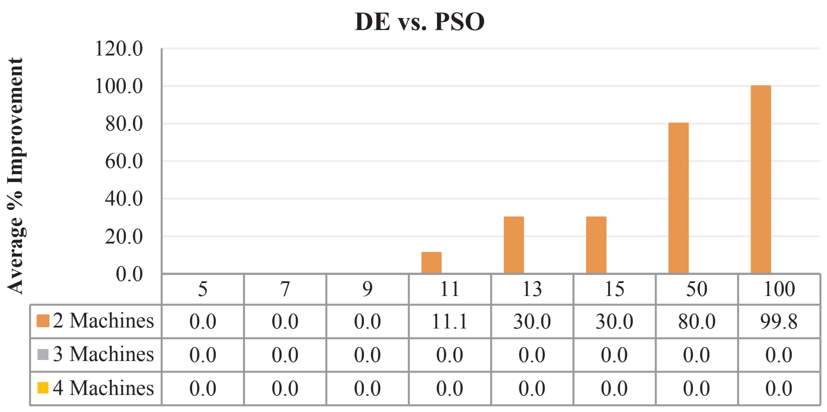
	5	7	9	11	13	15	50	100
🟧 2 Machines	18.0	37.7	51.2	45.8	53.0	74.3	82.6	71.6
⬜ 3 Machines	35.2	12.7	-2.0	90.2	100.0	89.8	99.9	99.6
🟨 4 Machines	35.6	4.1	0.0	8.3	0.0	10.0	20.0	0.0

Fig. 15. Experimental results of DE vs. PSO with a due date adjustment factor of 0.33.



	5	7	9	11	13	15	50	100
🟧 2 Machines	0.0	0.0	0.0	0.0	0.0	0.0	0.0	99.9
⬜ 3 Machines	0.0	0.0	0.0	0.0	0.0	0.0	0.0	33.3
🟨 4 Machines	0.0	0.0	0.0	0.0	0.0	0.0	0.0	10.0

Fig. 16. Experimental results of IBM ILOG CPLEX vs. PSO with a due date adjustment factor of 0.5.



	5	7	9	11	13	15	50	100
🟧 2 Machines	0.0	0.0	0.0	11.1	30.0	30.0	80.0	99.8
⬜ 3 Machines	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
🟨 4 Machines	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Fig. 17. Experimental results of DE vs. PSO with a due date adjustment factor of 0.5.

```
(17))
      Update position $X_{ij}^t$ (using Eq. (18))
      Develop a schedule using the heuristic discussed in Section 5.1 and
        determine the TWT
      if $f(X_i^t) < f(Pbest_i)$ then
        Update $Pbest_i$
      end if
      if $f(Pbest_i) < f(Gbest)$ then
        Update $Gbest$
      end if
    end for
    $t \leftarrow t + 1$
  end while
END
```

6. Computational experiments and results

Table 2 shows the parameters used to generate the problem instances. Ten instances were generated for each combination of number of jobs and due date adjustment factor (γ). Values of γ close to 0 indicate that the due dates are tight and values close to 1 indicate that the due dates are loose. Altogether, there were 240 experimental runs to test the solution quality of PSO. The proposed PSO algorithm was implemented in Matlab 7.1. In order to evaluate the proposed PSO algorithm, the solution obtained from PSO (i.e. TWT) and the run time required to solve the problem instances were compared to a discrete differential evolution algorithm (DE) and a commercial solver (i.e. IBM ILOG CPLEX) that was used to solve the model presented in Section 4. The percentage of improvement was calculated using Eq. (21). IBM ILOG CPLEX required long run times to solve the problem instances with 13, 15, 50, and 100 jobs; therefore, IBM ILOG CPLEX was allowed to run for 1800 s for those instances, and the best known solution was used for comparison.

where

$$z_j = \text{Discrete Uniform}\left[\mu\cdot\left(1-\frac{R}{2}\right),\mu\cdot\left(1+\frac{R}{2}\right)\right] \quad (19)$$

$$\mu = (1-T)\cdot C_{max}^{*} \quad (20)$$

C_{max}^{*} is equal to the makespan obtained when Full Batch LPT rule is applied. FBLPT rule is a polynomial time heuristic (Lee & Uzsoy, 1999).

$$\%\ Improvement = \frac{[\sum_{j\in J} w_j T_j(\text{CPLEX}) - \sum_{j\in J} w_j T_j(\text{PSO})]}{\sum_{j\in J} w_j T_j(\text{CPLEX})} \quad (21)$$

Table 2
Parameters to generate the problem instances.

Parameters	Description	Values
n	Number of jobs (problem size)	5, 7, 9, 11, 13, 15, 50, 100
m	Number of machines	2, 3, 4
s_m	Machine capacity	{40, 45, 50, 55}
Q	Interval for job sizes	Q = [1,30]
R	Due date tightness	R = 0.5
T	Due data spread out	T = 0.3
p_j	Job processing times	Discrete uniform [0, 48]
w_j	Job priorities	Discrete uniform [8, 48]
d_j	Job due dates	$\gamma.\ (p_j + z_j)$
γ	Due date adjustment factor	{0.2,0.33,0.5}

Table 3
Levels used to fine-tune PSO parameters.

Parameter	Levels
c_1	{0.5, 1, 2}
c_2	{0.5, 1, 2}
w	{0.6, 0.9, 1.2}
n_{pop}	{100, 200}
I	{100, 200}

Table 4
Average run time with a due date factor of 0.2.

Jobs	2 Machines			3 Machines			4 Machines		
	PSO	CPLEX	DE	PSO	CPLEX	DE	PSO	CPLEX	DE
5	7.9	0.5	12.56	8.1	0.4	13.33	8.7	0.8	14.20
7	10.4	5.6	13.27	11.8	6.7	13.54	12.5	79.8	14.63
9	13.6	84.4	14.73	15.9	266.8	13.95	9.8	417.4	15.46
11	17.7	1005.3	15.57	20.7	1345.9	14.17	22.8	1800.2	15.57
13	21.4	1799.0	16.53	23.8	1800.3	14.27	16.5	1136.7	15.71
15	26.2	1661.5	17.90	28.6	1800.5	14.58	24.0	1314.3	16.14
50	296.8	1806.5	37.83	284.7	1806.8	18.07	275.9	1750.0	19.93
100	170.9	1800.4	67.24	474.0	1800.6	22.53	477.5	1800.9	24.44

Table 5
Average run time with a due date factor of 0.33.

Jobs	2 Machines			3 Machines			4 Machines		
	PSO	CPLEX	DE	PSO	CPLEX	DE	PSO	CPLEX	DE
5	4.8	0.6	12.48	2.1	0.3	13.46	2.2	0.3	14.89
7	9.2	1.1	13.14	4.3	0.6	13.73	5.2	1.5	15.54
9	8.5	8.3	14.61	10.0	2.1	14.05	6.3	1.0	16.57
11	15.5	270.0	15.48	8.3	6.7	14.13	11.9	0.9	16.95
13	18.0	1329.3	16.58	0.2	0.6	14.35	0.2	0.5	17.45
15	22.3	1656.9	18.17	5.6	180.5	14.60	0.2	0.6	18.17
50	255.7	1803.6	38.29	82.2	539.5	17.96	1.2	50.9	22.38
100	430.1	1800.3	66.96	223.4	1800.2	22.51	3.7	1512.6	26.80

Two preliminary analyses were conducted to fine-tune the PSO parameters c_1 and c_2 (i.e. cognitive and social learning coefficients), inertia weight (w), the population size (n_{pop}), and the number of iterations (I). Three hundred twenty-four different experiments were conducted to fine-tune the PSO parameters for small problem instances (i.e. 5, 7, 9, 11, 13, and 15 jobs) and one hundred sixty-two different experiments were conducted for large problem instances (i.e. 50 and 100 jobs). The Minitab 17 statistical software was used to perform the statistical analyses. The levels for each one of the aforementioned PSO parameters are provided in Table 3.

Figs. 2–6 present the results from the statistical tests conducted

Table 6
Average run time with a due date factor of 0.5.

Jobs	2 Machines			3 Machines			4 Machines		
	PSO	CPLEX	DE	PSO	CPLEX	DE	PSO	CPLEX	DE
5	0.1	0.1	12.70	0.1	0.1	13.20	0.2	0.1	13.40
7	0.1	0.1	13.63	0.4	0.1	13.91	0.1	0.2	14.53
9	0.1	0.1	14.96	0.1	0.2	14.28	0.1	0.2	15.80
11	0.2	0.4	15.46	0.2	0.3	14.43	0.2	0.4	16.14
13	0.2	0.3	16.76	0.2	0.3	14.63	0.2	0.4	16.34
15	0.3	0.5	18.19	0.2	0.5	14.84	0.2	0.8	16.74
50	29.2	311.1	37.50	1.2	27.8	18.09	1.2	35.6	20.10
100	115.4	1800.4	66.29	3.8	1080.0	22.58	3.9	753.5	24.80

while fine tuning the PSO parameters for smaller problem instances. Based on the statistical analysis (i.e. analysis of variance), at a 0.05 level of significance, the factors affecting significantly the quality of the solution are the cognitive coefficient c_1, the cognitive and social learning c_2, and the inertia weight w. Considering the significance of the factors and the % improvement, the best set of parameters for the proposed PSO algorithm to solve smaller problem instances was selected as $c_1 = 2$, $c_2 = 2$, $w = 1.2$, $n_{pop} = 200$, and $I = 100$.

Figs. 7–11 show the statistical analysis for fine tuning PSO parameters for larger problem instances. According to the results, at a 0.05 level of significance, all the factors were affecting significantly the quality of the solution. The PSO parameters were set to $c_1 = 1$, $c_2 = 1$, $w = 0.6$, $n_{pop} = 200$, and $I = 200$ for 50 jobs and to $c_1 = 1$, $c_2 = 1$, $w = 0.6$, $n_{pop} = 200$, and $I = 100$ for 100 jobs. The decrement factor (α) was set to 0.99 for both small and large problem instances based on several experiments that yielded better results when compared to other values.

A similar statistical analysis was conducted in Amouie (2014) to determine the best parameters for DE. This algorithm is controlled by three main parameters: scaling parameter *F*, selection probability *CR* (also known as *pcr*), and population size *nPop*. A set of experiments were designed considering four different levels for parameters *CR* and *nPop*, and three different levels for *F*. Based on interaction plots and main effects plots for factors, the best set of parameters was selected as $F = 0.2$, $CR = 0.1$, and $nPop = 4\ 0$.

Figs. 12 and 13 summarize the average% of improvement in TWT with due date adjustment factor of 0.2 for instances with 2, 3, and 4 machines. From Fig. 12 it can be inferred that IBM ILOG CPLEX reports better solutions than the PSO algorithm for smaller problem instances. However, on 100-job problem instances the solutions from PSO outperformed the solutions from IBM ILOG CPLEX (on average between 78.3% and 95.1%). When PSO is compared with DE (see Fig. 13), the proposed PSO algorithm consistently outperforms DE on almost all problem instances.

Figs. 14 and 15 present the results with a due date factor of 0.33. On 2-machine problem instances the results from IBM ILOG CPLEX were superior to the PSO results for 5, 7, 11, 13, 15, and 50-job problem instances, whereas PSO outperformed IBM ILOG CPLEX by 90.3% on 100-job problem instances. On 3- and 4-machine problem instances the results from PSO matched the IBM ILOG CPLEX results for 5, 7, 9, 11, 13, 15, and 50-job problem instances. On 100-job problem instances PSO outperformed IBM ILOG CPLEX (on average between 80 and 100%). The PSO results were far better than DE on almost all problem instances with 2, 3, and 4 machine settings (see Fig. 15).

Figs. 16 and 17 present the results with a due date factor of 0.5. The PSO algorithm and IBM ILOG CPLEX yielded same results for 5, 7, 9, 11, 13, 15, and 50-job problem instances. PSO outperformed IBM ILOG CPLEX by at least 10% on 100-job problem instances. In Fig. 17 it can be seen that the proposed PSO algorithm outperforms DE on 11, 13, 15, 50, and 100-job problem instances with two machines and yields same results with three and four machines.

In terms of the experimental run times, the IBM ILOG CPLEX solver was restricted to run for a maximum of 1800 s (30 min) as it failed to converge to optimum even after running for several hours for larger problem instances (i.e. 50 and 100 jobs). It is evident from Tables 4–6 that PSO required shorter run time when compared to IBM ILOG CPLEX. On average, DE was much faster than PSO and IBM ILOG CPLEX. However, PSO required shorter run time compared to DE on 4 machine problem instances with 0.33 and 0.5 due date adjustment factors. Furthermore, the quality of the solution from PSO is far superior to DE on almost all the problem instances. The commercial solver took lesser time for five, seven and some nine-job instances. However, as the problem instances grew in size (with respect to the number of jobs) IBM ILOG CPLEX took longer times.

7. Conclusions

This paper considered a real world scheduling problem commonly observed in electronics manufacturing facilities. The problem under study required jobs to be grouped to process them as a batch on a set of non-identical batch processing machines. As the problem under study was NP-hard, a PSO algorithm was proposed. The PSO algorithm was implemented in Matlab and an experimental study was conducted to evaluate its performance – in terms of solution quality and run time. The proposed PSO algorithm's results were compared with a DE algorithm and a commercial solved used to solve the model discussed in Section 4.

On smaller problem instances with due date adjustment factors of 0.2 and 0.33, the commercial solver outperformed PSO algorithm. However, on 100-job problem instances the PSO algorithm outperformed the commercial solver. On smaller problem instances with due date adjustment factor of 0.5, the solutions obtained from PSO and the commercial solver were comparable. However, PSO solution was better than the commercial solver on the 100-job problem instances. The PSO algorithm consistently outperformed DE approach on all problem instances. When comparing the three approaches based on run time, the commercial solver quick to solve five and seven-job instances. However, as the problem instances grew in size (with respect to the number of jobs) the solver required longer run times. The PSO algorithm reported good quality solution for large problem instances in acceptable time. Our experimental study helps to conclude that the commercial solver is sufficient to solve smaller problem instances and the proposed PSO approach can be applied for solving more challenging problem instances with larger number of jobs. The scheduler at the electronics manufacturing facility, who motivated this study, was required to schedule 100–200 jobs per day on 3 or 4 machines. Our experimental study indicates that the proposed PSO algorithm is effective to solve large problem instances (i.e. 100 or more jobs). Consequently, the study was useful to convince the company to adopt the PSO algorithm to schedule their batch processing machines on a daily basis. Our research group is now working with the company to extend the PSO algorithm to consider jobs with non-zero release times and other objectives such as minimizing the number of tardy jobs.

References

Amouie, M. (2014). *Minimizing total weighted tardiness for non-identical parallel batch processing machines.* DeKalb, Illinois: Northern Illinois University.

Anghinolfi, D., & Paolucci, M. (2009). A new discrete particle swarm optimization approach for the single-machine total weighted tardiness scheduling problem with sequence-dependent setup times. *European Journal of Operational Research, 193*, 73–85.

Balasubramanian, H., Mönch, L., Fowler, J., & Pfund, M. (2004). Genetic algorithm based scheduling of parallel batch machines with incompatible job families to minimize total weighted tardiness. *International Journal of Production Research, 42*(8), 1621–1638.

Brucker, P., Mikhail, K., & Shafransky, Y. (1998). Batch scheduling with deadlines on parallel machines. *Annals of Operations Research, 83*, 23–40.

Chandru, V., Lee, C., & Uzsoy, R. (1993). Minimizing total completion on a batch processing machine with job families. *Operation Research Letters, 13*(2), 61–65.

Chang, P.-Y., Damodaran, P., & Melouk, S. (2004). Minimizing makespan on parallel batch processing machines. *International Journal of Production, 42*(19), 4211–4220.

Chang, J. -L., Chen, Y., Ma, X. -P. (2013). A hybrid particle swarm optimization algorithm for parallel batch processing machines scheduling. In *13th UK workshop on computational intelligence (UKCI)*, Guildford, 9–11 Sept. 2013.

Chiang, T.-C., Cheng, H.-C., & Fu, L.-C. (2010). A memetic algorithm for minimizing total weighted tardiness on parallel batch machines with incompatible job families and dynamic job arrival. *Computers & Operations Research, 37*(12), 2257–2269.

Damodaran, P., Diyadawagamage, D., Ghrayeb, O., & Vélez-Gallego, M. (2012). A particle swarm optimization algorithm for minimizing makespan of nonidentical parallel batch processing machines. *International Journal of Advanced Manufacturing Technology, 58*, 1131–1140.

Damodaran, P., Ghrayeb, O., & Guttikonda, M. (2013). GRASP to minimize makespan for a capacitated batch-processing machine. *Journal of Advanced Manufacturing Technology, 68*, 407–414.

Damodaran, P., Manjeshwar, P., & Srihari, K. (2006). Minimizing makespan on a batch processing machine with non-identical job sizes using genetic algorithms. *International Journal of Production Economics, 103*(2), 882–891.

Damodaran, P., Rao, A., & Mestry, S. (2013). Particle swarm optimization for scheduling batch processing machines in a permutation flowshop. *The International Journal of Advanced Manufacturing Technology, 64*, 989–1000.

Damodaran, P., & Vélez-Gallego, M. (2012). A simulated annealing algorithm to minimize makespan of parallel batch processing machines with unequal job ready times. *Expert Systems with Applications, 39*, 1451–1458.

Damodaran, P., Velez-Gallego, M., & Maya, J. (2009). A GRASP approach for makespan minimization on parallel batch processing machines. *Journal of Intelligent Manufacturing, 22*(5), 767–777.

Gokhale, R., & Mathirajan, M. (2014). Minimizing total weighted tardiness on heterogeneous batch processors with incompatible job families. *The International Journal of Advanced Manufacturing Technology, 70*, 1563–1578.

Graham, R., Lawler, E., Lenstra, J., & Kan, A. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics,* 287–326.

Ikura, Y., & Gimple, M. (1986). Efficient scheduling algorithms for a single batch processing machine. *Operations Research Letters, 5*(2), 61–65.

Kanagaa, E., & Valarmathi, M. (2012). Multi-agent based patient scheduling using particle swarm optimization. *Procedia Engineering, 30*, 386–393.

Kashan, A., & Karimi, B. (2008). Scheduling a single batch-processing machine with arbitrary job sizes and incompatible job families: An ant colony framework. *Journal of the Operational Research Society, 59*, 1269–1280.

Kashan, A. H., & Karimi, B. (2009). A discrete particle swarm optimization algorithm for scheduling parallel machines. *Computers & Industrial Engineering, 56*, 216–223.

Kennedy, J., Eberhart, R. (1995). Particle swarm optimization. In *IEEE international conference on neural networks*, Piscataway, NJ, USA.

Kennedy, J., Eberhart, R. (1997). A discrete binary version of the particle swarm algorithm. In *Proceedings of IEEE conference on systems, man, and cybernetics*. Piscataway, NJ, USA.

Kennedy, J., Eberhart, R. C., & Shi, Y. (2001). *Swarm intelligence. vol. 1*, San Francisco: Kaufmann700–720.

Kurz, M. (2003). On the structure of optimal schedules for minimizing total weighted tardiness on parallel batch-processing machines. In *IIE annual conference*.

Lawler, E. (1977). A "Pseudopolynomial" algorithm for sequencing jobs to minimize total tardiness. *Annals of Discrete Mathematics, 1*, 331–342.

Lee, C.-Y., & Uzsoy, R. (1999). Minimizing makespan on a single batch processing machine with dynamic job arrivals. *International Journal of Production Research, 37*(1), 219–236.

Lee, C.-Y., Uzsoy, R., & Martin-Vega, L. (1992). Efficient algorithms for scheduling semiconductor burn-in operations. *Operations Research, 40*(4), 764–775.

Liao, C.-J., Tseng, C.-T., & Luarn, P. (2007). A discrete version of particle swarm optimization for flowshop scheduling problems. *Computers & Operations Research, 34*, 3099–3111.

Lin, Y.-K. (2013). Particle swarm optimization algorithm for unrelated parallel machine scheduling with release dates. *Mathematical Problems in Engineering, 2013*, 9.

Liu, L., Ng, C., & Cheng, T. (2009). Scheduling jobs with release dates on parallel batch processing machines. *Discrete Applied Mathematics, 157*, 1825–1830.

Liu, L., Ng, C., & Cheng, T. (2014). Scheduling jobs with release dates on parallel batch processing machines to minimize the makespan. *Optimization Letters, 8*(1), 307–318.

Malve, S., & Uszoy, R. (2007). A genetic algorithm for minimizing maximum lateness on parallel identical batch processing machines with dynamic job arrivals and incompatible job families. *Computers & Operations Research, 34*, 3016–3028.

Mönch, L., Balasubramanian, H., Fowler, J. W., Pfund, M. E. (2003). Minimizing total weighted tardiness on parallel batch process machines using genetic algorithms. In *Operations research proceedings*.

Niu, Q., Zhou, T., & Wang, L. (2010). A hybrid particle swarm optimization for parallel machine total tardiness scheduling. *International Journal of Advanced Manufacturing Technology, 49*, 723–739.

Nong, Q., Cheng, T., & Ng, C. (2008). An improved on-line algorithm for scheduling on two unrestrictive parallel batch processing machines. *Operations Research Letters, 36*(5), 584–588.

Pan, Q.-K., Tasgetiren, M. F., & Liang, Y.-C. (2008). A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. *Computers & Operations Research, 35*, 2807–2839.

Potts, C., & Kovalyov, M. (2000). Scheduling with batching: A review. *European Journal of Operational Research, 120*, 228–249.

Raghavan, N., & Venkataramana, M. (2009). Parallel processor scheduling for minimizing total weighted tardiness using ant colony optimization. *The International Journal of Advanced Manufacturing Technology, 41*, 986–996.

Shao, X., Weiqi, L., Liu, Q., & Zhang, Q. (2013). Hybrid discrete particle swarm optimization for multi-objective flexible job-shop scheduling problem. *The International Journal of Advanced Manufacturing Technology, 67*, 2885–2901.

Shi, Y., Eberhart, R. (1998). A modified particle swarm optimizer. In *Evolutionary computation proceedings*. IEEE World Congress on Computational Intelligence.

Tang, L., & Wang, X. (2013). An improved particle swarm optimization algorithm for the hybrid flowshop scheduling to minimize total weighted completion time in process industry. *IEEE Transactions on Control Systems Technology, 18*(6), 1303–1314.

Tasgetiren, M. F., Liang, Y. -C., Sevkli, M., Gencyilmaz, G. (2004). Particle swarm optimization algorithm for single machine total weighted tardiness. In *Proceedings of the 2004 congress on evolutionary computation, CEC2004*.

Tasgetiren, M. F., Liang, Y.-C., Sevkli, M., & Gencyilmaz, G. (2007). A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. *European Journal of Operational Research, 177*, 1930–1947.

Uzsoy, R. (1995). Scheduling batch processing machines with incompatibles job families. *International Journal of Production Research, 33*(10), 2685–2708.

Wang, C., & Uzsoy, R. (2002). A genetic algorithm to minimize maximum lateness on a batch processing machine. *Computers & Operations Research, 29*, 1621–1640.

Yang, C.-I., Chou, J.-H., & Chang, C.-K. (2014). Hybrid taguchi-based particle swarm optimization for flowshop. *Arabian Journal for Science and Engineering, 39*, 2393–2412.

Yuan, J., Ng, C., & Cheng, T. (2011). Best semi-online algorithms for unbounded parallel batch scheduling. *Discrete Applied Mathematics, 159*(8), 838–847.