

Minimizing makespan in a no-wait flowshop with two batch processing machines using estimation of distribution algorithm

Shengchao Zhou^{a*}, Xueping Li^b, Huaping Chen^c and Cong Guo^b

^a*School of Management, University of Science and Technology of China, Hefei, China;* ^b*Department of Industrial and Systems Engineering, University of Tennessee, Knoxville, USA;* ^c*School of Computer Science and Technology, University of Science and Technology of China, Hefei, China*

(Received 24 September 2015; accepted 5 January 2016)

This paper studies the problem of minimising makespan in a no-wait flowshop with two batch processing machines (comprised of a parallel batch processing machine and a serial batch processing machine), non-identical job sizes and unequal ready times. We propose a population-based evolutionary method named estimation of distribution algorithm (EDA). Firstly, the individuals in the population are coded into job sequences. Then, a probabilistic model is built to generate new population and an incremental learning method is developed to update the probabilistic model. Thirdly, the best-fit heuristic is used to group jobs into batches and a least idle/waiting time approach is proposed to sequence the batches on batch processing machines. In addition, some problem-dependent local search heuristics are incorporated into the EDA to further improve the searching quality. Computational simulation and comparisons with some existing algorithms demonstrate the effectiveness and robustness of the proposed algorithm. Furthermore, the effectiveness of embedding the local search method in the EDA is also evaluated.

Keywords: scheduling; no-wait flowshop; batch processing machines; makespan; estimation of distribution algorithm (EDA)

1. Introduction

Batch processing machines (BPMs) can process a number of jobs in the form of a batch. BPMs have wide applications in many industrial environments such as electronics manufacturing facilities (Lee, Uzsoy, and Martin-Vega 1992; Zhou et al. 2013; Jia, Jiang, and Li 2015), tire production plants (Oulamara, Finke, and Kamgaing Kuiteing 2009), casting industry (Mathirajan and Sivakumar 2006) and metal working industry (Tang and Liu 2009a; Gong, Tang, and Duin 2010; Tang et al. 2014). BPMs can be classified into serial and parallel batching machines. A serial batching (i.e. s-batch) machine processes the jobs in a batch serially (Cheng and Wang 1998; Glass, Potts, and Strusevich 2001). A parallel batching (i.e. p-batch) machine can simultaneously process several jobs (Ahmadi et al. 1992; Rossi, Pandolfi, and Lanzetta 2014; Li, Ishii, and Chen 2015). There are basically two types of BPM scheduling problems: problems with fixed or varying batch processing times. In fixed batch processing time problems (Sung, Kim, and Yoon 2000; Lin and Liao 2013), the processing time of a batch is independent of the jobs belonging to the batch as a constant. In varying batch processing time problems (Potts, Strusevich, and Tautenhahn 2001; Behnamian et al. 2012; Abedi et al. 2015), however, the processing time of a batch is dependent on the jobs in the batch. For example, the processing time of a batch can be equal to the sum of the processing times of all jobs belonging to the batch or the longest processing time of all jobs in the batch.

This paper considers the two-machine flowshop scheduling problem with BPMs, a no-wait constraint, non-identical job sizes and unequal ready times. The first BPM is a parallel BPM and the second BPM is a serial BPM. Batches of jobs cannot wait between the two machines. Hence, a batch that completes processing on the first machine has to remain on that machine if the second machine is busy processing another batch. A constant set-up time is considered on the second machine before each batch is processed on it. Additionally, we assume that jobs arrive dynamically at the first machine. The objective is to group the jobs into batches and schedule the batches such that the makespan (i.e. the completion time of the last batch leaving the flowshop) is minimised. A smaller makespan usually implies a higher utilisation of the production system.

This problem under study is motivated by problems arise in industrial operations. For example, for a cleaning test operation observed at a sensor manufacturing facility, the substrates (or jobs) need to be cleaned to remove any residuals. The cleaning test takes place in two baths (or processing machines). Multiple substrates with different sizes can be cleaned at one time in the first bath as long as the total size of all the substrates in a batch does not exceed the bath capacity. The time taken here depends on the composition of the batch. In general, the cleaning time is equal to the longest cleaning time among

*Corresponding author. Email: zhousc@mail.ustc.edu.cn

all the substrates in the batch. This cleaning time of each substrate can be estimated a priori. The second bath can also test several substrates at a time but here the substrates are handled sequentially. Consequently, the time taken to wash in this bath is equal to the sum of the cleaning times of all substrates in the batch. One of the considerations in scheduling the batches is that each batch should be continuously processed from the start to the end (i.e. no-wait constraint). For more details of this process, please refer to [Muthuswamy et al. \(2012\)](#).

The rest of this paper is organised as follows. Section 2 presents the literature review of two-stage flowshop scheduling problems with BPMs and some applications of estimation of distribution algorithms (EDA) in the literature. Problem definition is given in Section 3. In Section 4, the EDA for our problem is proposed in detail. Section 5 presents the EDA with local search. Computational results and comparisons are provided in Section 6. Some concluding remarks are summarised in Section 7.

2. Literature review

2.1 Two-stage flowshops with BPMs

Problems related to scheduling in a flowshop with BPMs have received considerable attention in recent years. [Ahmadi et al. \(1992\)](#) considered the problem of scheduling a two-stage flowshop in which one or both of the stages may comprise a BPM. The processing time of a batch is a constant regardless of the number of jobs contained in it. They studied the complexity analysis for two problem instances with the objective of makespan and total completion time. [Cheng and Wang \(1998\)](#) considered a class of batching and scheduling problems in the two-machine flowshop where one machine is a discrete processor and the other one is a BPM. The batch processing time is equal to the total processing time of the jobs contained in that batch. A constant set-up time is incurred whenever a batch is formed on the BPM. All problems were shown to be NP-complete in the ordinary sense. [Cheng, Lin, and Toker \(2000\)](#) considered the two-machine flowshop where both machines process jobs in batches. They proved the strong NP-hardness of the problem, presented properties and polynomial algorithms for some special cases, and proposed heuristic algorithms to deal with the general problem.

[Sung and Min \(2001\)](#) considered the problem of scheduling one or two BPMs in a flowshop environment. The processing times of batches are independent of the composition of the batch and identical. The objective is to minimise the earliness/tardiness (E/T) measure. [Sung and Kim \(2002\)](#) extended the two-machine flowshop of [Ahmadi et al. \(1992\)](#) to the situation in which the jobs arrive dynamically at the first machine. The problem was proven as strongly NP-hard and an efficient heuristic was presented. [Sung and Kim \(2003\)](#) analysed a two-machine flowshop comprising BPMs with respect to three due date-related problems. The BPMs can process jobs simultaneously as long as the number of jobs in the batch is less than a predetermined number.

[Su \(2003\)](#) examined a two-stage flowshop scheduling problem with limited waiting time constraints. There is a BPM in the first stage and a discrete machine in the second stage. A mixed integer programming formulation and a heuristic were developed to solve the problem. [Glass, Potts, and Strusevich \(2001\)](#) studied a problem of batching and scheduling on two BPMs in a flowshop environment. The processing time of a batch is the sum of the processing times of the jobs in that batch. They did not consider any restrictions on the number of jobs that can be assigned to a batch. An integer programming formulation and a heuristic were developed to minimise the makespan.

[Lin and Cheng \(2001\)](#) investigated flowshops with two serial BPMs and a no-wait constraint. A constant set-up time is considered before a batch is processed on each machine. They showed that the problem of minimising the completion time is strongly NP-hard. [Lin and Cheng \(2005\)](#) examined a two-machine flowshop scheduling problem where a discrete machine is followed by a BPM. The jobs have identical sizes and no restrictions are imposed on the number of jobs that can be assigned to a batch. The processing time of a batch is equal to a constant set-up time plus the sum of the processing times of all the jobs in that batch. They showed that the problem is strongly NP-hard and designed some heuristics for deriving approximate solutions.

[Oulamara, Kovalyov, and Finke \(2005\)](#) studied a no-wait flowshop problem with two parallel BPMs and proposed a polynomial algorithm for the problem. [Oulamara, Finke, and Kamgaing Kuiteing \(2009\)](#) investigated a flowshop scheduling problem with a BPM and job compatibilities. They showed that minimising the makespan is NP-hard in the ordinary sense and presented three heuristics to minimise the makespan. [Tang and Liu \(2009b\)](#) studied the two-machine flowshop scheduling problem with batching and release times, in which the objective is to minimise the makespan. They formulated the problem as a mixed integer programming model and showed that it is strongly NP-hard. A dynamic programming-based heuristic was developed. [Gong, Tang, and Duin \(2010\)](#) considered a two-stage flowshop scheduling problem on a BPM and a discrete machine with blocking and shared set-up times. For the objective of minimising the makespan, they proved that the problem is strongly NP-hard.

All of the research works mentioned above assume that the jobs have identical sizes and the capacity of a BPM can be defined by the number of jobs in a batch or is unbounded. Moreover, these works only consider one type of BPMs, either

Table 1. The literature related to two-stage flowshop scheduling problems with BPM(s).

References	Job size	Ready time	Number of BPMs	Type of BPMs	Batch processing time	Performance criterion
Ahmadi et al. (1992)	I	Equal	One or two	–	Constant	$C_{\max}, \sum C_j$
Cheng and Wang (1998)	I	Equal	One	s-batch	Sum	C_{\max}
Cheng, Lin, and Toker (2000)	I	Equal	Two	s-batch	Sum	C_{\max}
Sung and Min (2001)	I	Equal	One or two	–	Constant	E/T
Glass, Potts, and Strusevich (2001)	I	Equal	Two	s-batch	Sum	C_{\max}
Lin and Cheng (2001)	I	Equal	Two	s-batch	Sum	C_{\max}
Sung and Kim (2002)	I	Unequal	One	–	Constant	C_{\max}
Sung and Kim (2003)	I	Equal	Two	–	Constant	$T_{\max}, N_T, \sum T_j$
Su (2003)	I	Equal	One	–	Constant	C_{\max}
Damodaran and Srihari (2004)	N	Equal	Two	p-batch	Longest	C_{\max}
Lin and Cheng (2005)	I	Equal	One	s-batch	Sum	C_{\max}
Oulamara, Kovalyov, and Finke (2005)	I	Equal	Two	p-batch	Longest	C_{\max}
Oulamara (2007)	I	Equal	Two	p-batch, s-batch	Longest, sum	C_{\max}
Liao and Liao (2008)	N	Equal	Two	p-batch	Longest	C_{\max}
Oulamara, Finke, and Kamgaing Kuiteing (2009)	I	Equal	One	p-batch	Longest	C_{\max}
Tang and Liu (2009b)	I	Unequal	One	s-batch	Sum	C_{\max}
Kashan and Karimi (2009)	N	Equal	Two	p-batch	Longest	C_{\max}
Mirsanei, Karimi, and Jolai (2009)	N	Equal	Two	p-batch	Longest	C_{\max}
Manjeshwar, Damodaran, and Srihari (2009)	N	Equal	Two	p-batch	Longest	C_{\max}
Jolai et al. (2009)	I	Equal	Two	p-batch, s-batch	Longest, sum	C_{\max}
Gong, Tang, and Duin (2010)	I	Equal	One	–	Constant	C_{\max}
Liao and Huang (2011)	N	Equal	Two	p-batch	Longest	C_{\max}
Manjeshwar, Damodaran, and Srihari (2011)	N	Equal	Two	p-batch	Longest	C_{\max}
Muthuswamy et al. (2012)	N	Unequal	Two	p-batch, s-batch	Longest, sum	C_{\max}
Chen et al. (2014)	N	Unequal	Two	p-batch	Longest	C_{\max}
Cheng et al. (2014)	N	Equal	Two	p-batch	Longest	C_{\max}
This paper	N	Unequal	Two	p-batch, s-batch	Longest, sum	C_{\max}

Note: 'I' represents identical; 'N' represents non-identical.

s-batch or p-batch machines, but not both. Considering jobs with non-identical sizes, Damodaran and Srihari (2004) presented two mixed integer formulations for scheduling two BPMs in a flowshop scenario when the intermediate buffer is infinite or zero. The processing time of a batch is the longest processing time among all the jobs in that batch on both machines. Liao and Liao (2008) and Kashan and Karimi (2009) investigated the same problem as Damodaran and Srihari (2004) and improved the models proposed by Damodaran and Srihari (2004). Mirsanei, Karimi, and Jolai (2009) and Manjeshwar, Damodaran, and Srihari (2009) developed two simulated annealing (SA) algorithms to minimise makespan in a two-machine flowshop with infinite buffer, respectively. Liao and Huang (2011) proposed a Tabu search heuristic and Manjeshwar, Damodaran, and Srihari (2011) proposed a genetic algorithm (GA) for the same case. Cheng et al. (2014) designed a polynomial time algorithm under the assumption that the processing time of a job on the first machine has a positive correlation with its processing time on the second machine.

This paper differs from the above research in that we consider the flowshop scheduling problem with a parallel BPM and a serial BPM instead of two parallel BPMs. Oulamara (2007) and Jolai et al. (2009) investigated the flowshop scheduling problem with both parallel and serial BPMs. However, they did not include unequal ready times and non-identical job sizes in their models. To the best of our knowledge, only Muthuswamy et al. (2012) examined a similar problem. In this work, the authors proposed a mathematical formulation and a particle swarm optimisation (PSO) algorithm.

The literature related to two-stage flowshop scheduling problems with BPM(s) is shown in Table 1.

2.2 Estimation of distribution algorithm

The EDA is a relatively new population-based evolutionary algorithm introduced by Muhlenbein and Paass (1996). Unlike other evolutionary algorithms, the EDA generates new solutions by sampling from a probabilistic model which characterises the solution space. According to different problem types, the EDA may employ different probabilistic models, including univariate, bivariate and multivariate probabilistic models. The univariate probabilistic models assume that there is no

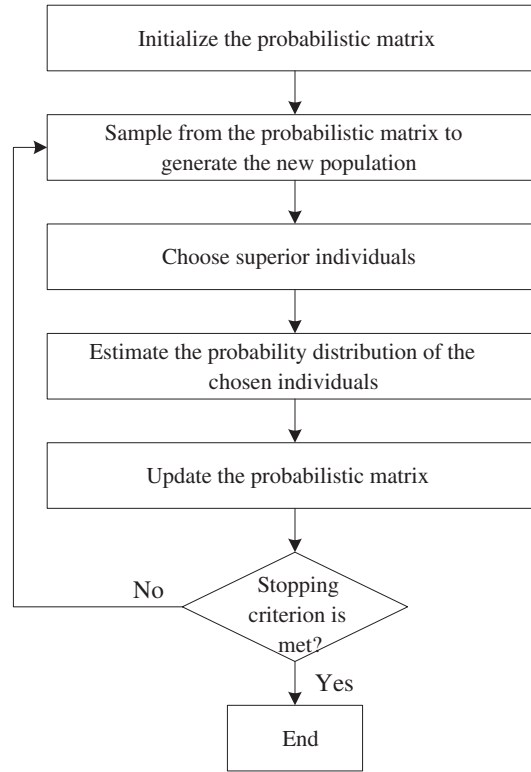


Figure 1. The general framework of the EDA.

interaction between/among variables such as the compact GA (Harik, Lobo, and Goldberg 1999), the self-guided GA (Chen, Chang et al. 2012), and the EDA (Wang et al. 2013). The bivariate probabilistic models consider pairwise variable interactions such as the mutual-information-maximising input clustering (De Bonet, Isbell, and Viola 1997), the combining optimisers with mutual information trees (Baluja and Davies 1997), the bivariate marginal distribution algorithm (Pelikan and Mühlenbein 1999) and the extended artificial chromosomes GA (Chen, Chen et al. 2012). Whereas the multivariate probabilistic models consider multivariate interactions such as the factorised distribution algorithms (Muhlenbein and Mahnig 1999), the extended compact GA (Harik 1999) and the Bayesian optimisation algorithm (Pelikan 2005).

The EDA is being widely used to solve various kinds of combinatorial optimisation problems such as permutation flowshop scheduling (Chang et al. 2009; Zhang and Li 2011; Pan and Ruiz 2012), nurse rostering (Aickelin, Burke, and Li 2006), software testing (Sagarna and Lozano 2005), single machine scheduling (Chang et al. 2010), jobshop scheduling (Wang et al. 2012; Liu, Fan, and Liu 2015) among many other applications. For a complete review of the EDA, please refer to Larrañaga and Lozano (2002), Pelikan, Goldberg, and Lobo (2002) and Lozano et al. (2006).

The EDA provides a new tool of stochastic optimisation methods (Wang et al. 2012). Different from the genetic algorithm which explicitly generates new offspring using crossover and mutation operators, the EDA generates new offspring implicitly. Firstly, the EDA builds a probabilistic model based on the statistical information extracted from previous searches. Then, the EDA samples from the probabilistic model and generates new solutions. Accordingly, the statistical information, as well as the probabilistic model, is updated using some of the new solutions. The general framework of the EDA is described in Figure 1.

3. Problem definition

In this section, the problem under consideration is formulated as a mixed integer programming model. The parameters and variables used are given as follows.

Parameters:

- J The set of all jobs, $J = \{1, 2, \dots, n\}$, where n is the total number of jobs.
- M The set of machines, $M = \{1, 2\}$.
- s_j The size of job j .

- p_{jm} The processing time of job j on machine m .
- r_j The ready time of job j .
- S The capacity of the first machine.
- T The set-up time on the second machine.

Decision variables:

- X_{jb} 1, if job j belongs to the b th batch; 0, otherwise.
- Y_b 1, if the b th batch is not empty; 0, otherwise.
- P_{bm} The processing time of the b th batch on machine m .
- R_b The ready time of the b th batch.
- U_{b1} The starting time of the b th batch processed on the first machine.
- D_{b1} The departure time of the b th batch processed on the first machine.
- C_{b2} The completion time of the b th batch processed on the second machine.
- C_{\max} The makespan.

The problem considered in this research can be formally stated as follows.

- (1) There are n jobs to be processed in a two-machine flowshop with BPMs. The jobs arrive dynamically at the first machine and the ready time of job j is denoted by r_j . Each job has a size s_j and the processing time of job j on machine m is denoted by p_{jm} .
- (2) The first machine has a capacity S and can process a batch of jobs as long as the total size of all the jobs in the batch does not exceed the capacity. The batch is ready for processing only after all the jobs in the batch are ready. The processing time of batch b on the first machine is equal to the longest processing time of the jobs in the batch. The processing time of batch b on the second machine is equal to the sum of processing times of the jobs in the batch. A constant set-up time T is considered on the second machine before each batch is processed on it. The sequence in which the batches are to be processed is the same for each machine.
- (3) Batch b cannot wait between the two machines, though it is completely processed on the first machine. In other words, the batch must stay an extra time on the first machine if the second machine is not free.
- (4) The objective is to minimise the makespan C_{\max} .

This problem can be denoted as $F_2|p - \text{batch}(1), s - \text{batch}(2), \text{no-wait}, s_j, r_j|C_{\max}$ according to the three-field notation (Graham et al. 1979). The mixed integer programming model for the problem is formulated as follows. A similar formulation can be found in Muthuswamy et al. (2012).

$$\text{Minimize } C_{\max} \quad (1)$$

Subject to :

$$\sum_{b=1}^n X_{jb} = 1, \quad j = 1, \dots, n \quad (2)$$

$$\sum_{j=1}^n s_j X_{jb} \leq S, \quad b = 1, \dots, n \quad (3)$$

$$P_{b1} \geq p_{j1} X_{jb}, \quad j = 1, \dots, n; b = 1, \dots, n \quad (4)$$

$$P_{b2} \geq \sum_{j=1}^n p_{j2} X_{jb}, \quad b = 1, \dots, n \quad (5)$$

$$U_{b1} \geq r_j X_{jb}, \quad j = 1, \dots, n; b = 1, \dots, n \quad (6)$$

$$U_{b1} \geq D_{b-1,1}, \quad b = 2, \dots, n \quad (7)$$

$$D_{b1} \geq U_{b1} + P_{b1}, \quad b = 1, \dots, n \quad (8)$$

$$D_{b1} \geq C_{b-1,2}, \quad b = 2, \dots, n \quad (9)$$

$$Y_b \geq X_{jb}, \quad j = 1, \dots, n; b = 1, \dots, n \quad (10)$$

$$C_{b2} \geq D_{b1} + T \times Y_b + P_{b2}, \quad b = 1, \dots, n \quad (11)$$

$$C_{\max} \geq C_{b2}, \quad b = 1, \dots, n \quad (12)$$

$$X_{jb} \in \{0, 1\}, \quad j = 1, \dots, n; b = 1, \dots, n \quad (13)$$

$$Y_b \in \{0, 1\}, \quad b = 1, \dots, n \quad (14)$$

Table 2. Illustration of the job sequence based representation.

Position k	1	2	3	4	5	6	7	8
x_k	4	7	2	8	6	3	1	5
Sequence	4	7	2	8	6	3	1	5

$$U_{b1}, D_{b1}, C_{b2} \geq 0, \quad b = 1, \dots, n \quad (15)$$

The objective (1) minimises the makespan. Constraint (2) ensures that each job is assigned to exactly one batch. Constraint (3) requires that the total size of all the jobs in a batch does not exceed the machine capacity. Constraint (4) ensures that the processing time of the b th batch on machine 1 is at least equal to the longest processing time of all the jobs in the batch. Constraint (5) determines the processing time on machine 2 for batch b . Constraint (6) indicates that the b th batch is processed on machine 1 only after all the jobs in the batch have arrived. Constraint (7) ensures that the starting time of the b th batch on machine 1 is at least the departure time of the $(b - 1)$ th batch on machine 1. Constraint (8) ensures that the departure time of the b th batch on machine 1 is at least equal to its starting time on machine 1 plus its processing time on machine 1. Constraint (9) guarantees that a batch cannot leave from machine 1 until the previous batch completes its processing on machine 2. Constraint (10) relates the Y_b and X_{jb} variables. Y_b is equal to 1 as long as the b th batch contains at least one job. Constraint (11) decides the completion time of each batch on machine 2. Constraint (12) determines the makespan. Finally, constraints (13)–(15) define the range of the variables.

4. The proposed EDA algorithm

In this section, we will present an EDA for the two-machine no-wait flowshop with BPMs, non-identical job sizes and unequal ready times. The solution representation and initial population, probabilistic model and updating mechanism, and batching and sequencing decisions are explained as follows.

4.1 Solution representation and initial population

The job sequence-based representation is widely used in the literature for two-stage flowshop scheduling problems (Muthuswamy et al. 2012; Chen et al. 2014). Therefore, it is also adopted in our EDA. In this representation, the i th number of the sequence denotes the job placed at position i . A numerical example is illustrated in Table 2.

In order to guarantee the initial population with a certain quality and diversity, we construct the initial population in two ways. Firstly, some prior rules, including Johnson's rule, the longest processing time (LPT) rule, the shortest processing time (SPT) rule and the first in first out (FIFO) rule, are employed to generate initial individuals. Johnson's rule is one of the well-known heuristics in the literature and it provides an optimal solution for makespan minimisation in a two-stage flowshop with discrete machines. The LPT rule sequences the jobs in descending order of the sum of their processing times on both the machines and the SPT rule sequences the jobs in ascending order of the sum of their processing times on the machines. The FIFO rule is applied to the jobs based on their ready times. Secondly, the rest of the individuals in the initial population is generated randomly. Finally, a population with Q individuals is created.


4.2 Probabilistic model and updating mechanism

As mentioned previously, the EDA generates new population by sampling from a probabilistic model which directly impacts the performance of the algorithm. Thus, the construction of the probabilistic model is a crucial step for excellently designing the EDA. In our algorithm, a probabilistic matrix is built to determine the estimation of distribution model as follows:

$$P(t) = \begin{pmatrix} p_{11}(t) & \dots & p_{1n}(t) \\ \vdots & \ddots & \vdots \\ p_{n1}(t) & \dots & p_{nn}(t) \end{pmatrix} \quad (16)$$

where $p_{jk}(t)$ denotes the probability of job j to be placed at position k at generation t . At the beginning of the algorithm, each $p_{jk}(0)$ is initialised to be $1/n$. This initialisation means that all solutions can be sampled uniformly.

Individual 1	2	3	4	1
Individual 2	4	1	3	2
Individual 3	1	2	3	4
Individual 4	1	3	4	2
Individual 5	4	1	3	2
Individual 6	1	4	2	3
Individual 7	4	1	2	3
Individual 8	2	3	1	4



		Position			
		1	2	3	4
Job	1	3/8	3/8	1/8	1/8
	2	2/8	1/8	2/8	3/8
	3	0	3/8	3/8	2/8
	4	3/8	1/8	2/8	2/8

Figure 2. Extracting statistical information and building a probabilistic matrix.

Table 3. The probability and accumulated probability of each job at position 1.

Job	Probability	Accumulated
1	3/8	3/8
2	2/8	5/8
3	0	5/8
4	3/8	1

		Position			
		1	2	3	4
Job	1	0	3/8	1/8	1/8
	2	0	0	0	0
	3	0	3/8	3/8	2/8
	4	0	1/8	2/8	2/8

Figure 3. The probabilistic matrix after assigning job 2 at position 1.

According to the probabilistic matrix, new individuals are generated by sampling at generation t . For each new individual, position k (from 1 to n) is assigned job j by proportional selection based on probability $p_{jk}(t)$. If a job is assigned to a certain position, the data in the probabilistic matrix for that job and that position would be set to zero. In such a way, a number of Q individuals are generated.

In order to illustrate how the new individual generation procedure works, a four-job instance is used. Suppose there are eight individuals which are selected to build the probabilistic matrix, as shown in the left-hand side of Figure 2. Then, we construct the probabilistic matrix P as shown in the right-hand side of Figure 2. To generate an individual, jobs are assigned to each position by proportional selection based on the probability of each job at this position. Taking position 1 as an example, the probability and accumulated probability of each job at the position is given in Table 3. Suppose job 2 is chosen to be assigned at position 1, and then the probabilistic matrix is updated as shown in Figure 3. Following the same procedure, all positions will be assigned a job and a new individual is generated.

After the new population is generated, the fitness of these individuals is evaluated and the best Q_1 individuals are chosen to update the probabilistic matrix, where $Q_1 = \alpha \times Q$, $0 < \alpha < 1$. The probabilistic matrix is updated by the following equation:

$$p_{jk}(t+1) = (1 - \beta)p_{jk}(t) + \frac{\beta}{Q_1} \sum_{l=1}^{Q_1} Z_{jk}^l, \quad j, k = 1, \dots, n \quad (17)$$

where $\beta \in (0, 1)$ is the learning rate and Z_{jk}^l is a binary variable in the l th individual which is defined by Equation (18). As can be seen in Equation (17), we consider both of historical knowledge $p_{jk}(t)$ and current knowledge $1/Q_1 \sum_{l=1}^{Q_1} Z_{jk}^l$ which is learnt from the superior individuals. The relative importance of the two kinds of knowledge is determined by rate β . This updating method is an incremental learning one as suggested in Baluja and Davies (1998).

$$Z_{jk}^l = \begin{cases} 1 & \text{if job } j \text{ is assigned to position } k \\ 0 & \text{otherwise} \end{cases} \quad (18)$$

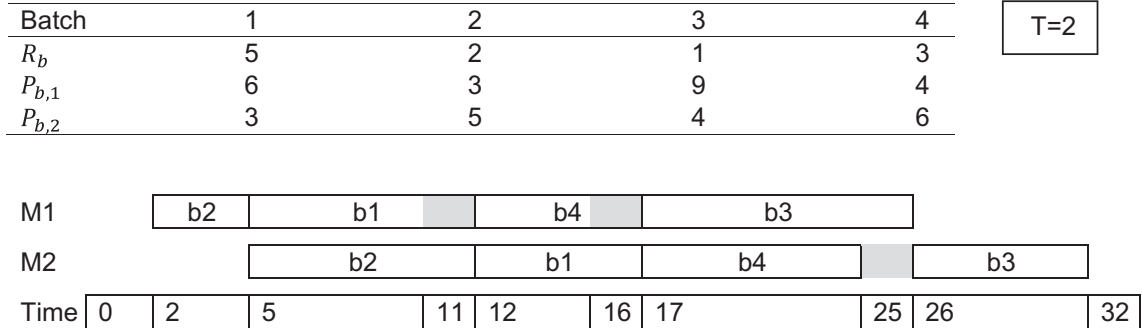


Figure 4. Illustration of the LIWT heuristic.

4.3 Batching decision

The best-fit (BF) heuristic is one of the well-known methods for the bin packing problem (Bramel and Simchi-Levi 1998). It has been applied to group jobs into batches for machine scheduling problems (Ghazvini and Dupont 1998; Damodaran, Kumar Manjeshwar, and Srihari 2006; Chen, Du, and Huang 2010). In our EDA, we also employ the BF heuristic to form batches. The BF adapted to this scheduling problem is as follows:

- Step 1 Sequence the jobs randomly.
- Step 2 Select the job at the head of the list and assign it to the batch with the least remaining capacity. If the job does not fit in any existing batches, create a new batch. Repeat step 2 until all the jobs have been assigned to a batch.

4.4 Sequencing decision

In order to sequence the formed batches processed on the BPMs, a method, which we refer to as the least idle/waiting time (LIWT) heuristic, is developed based on the scheduling heuristic of Muthuswamy et al. (2012). Firstly, we define the idle/waiting time of the b th batch on machine m by θ_{bm} and the total idle/waiting time of the b th batch on both the machines by θ_b . Then θ_{bm} and θ_b can be determined by the following equations:

$$\begin{cases} D_{b+1,1} = \max\{\max\{R_{b+1}, D_{b1}\} + P_{b+1,1}, C_{b2}\} \\ \theta_{b+1,1} = D_{b+1,1} - (D_{b1} + P_{b+1,1}) \end{cases} \quad (19)$$

$$\begin{cases} C_{b+1,2} = D_{b+1,1} + T + P_{b+1,2} \\ \theta_{b+1,2} = C_{b+1,2} - (C_{b2} + T + P_{b+1,2}) \end{cases} \quad (20)$$

$$\theta_{b+1} = \theta_{b+1,1} + \theta_{b+1,2} \quad (21)$$

The LIWT heuristic is described as follows:

- Step 1 Arrange the batch with the least sum of the batch ready time and the batch processing time on machine 1 (i.e. $R_b + P_{b1}$) as the first batch.
- Step 2 Calculate the total idle/waiting times θ_{b+1} of all the unscheduled batches on both the machines by Equations (19)–(21), and schedule the batch with the least idle/waiting time as the next batch. Repeat this step until all the batches have been scheduled on the machines.

An example of 4 batches is used to illustrate the LIWT heuristic. The ready times and processing times of the batches, the set-up time and their schedule are given in Figure 4. The shadow areas in the figure show the idle/waiting time.

5. The EDA with local search

5.1 Local search heuristics

The EDA makes effective use of global statistic information rather than local information about individual solutions during the evolution process. An efficient EDA should find an appropriate balance between global statistic information and local information. Hence in this paper, some problem-dependent local search heuristics are proposed and incorporated into the EDA in order to further improve the solution quality. The local search is designed for the batch sequencing phrase and applied to the best solution at each generation. Two types of neighbourhood structures are considered, namely insert-based local

search and swap-based local search. For a given sequence of batches, it tries to reduce the makespan through inserting a batch into another position or exchanging two batches at different positions. The pseudo code of the local search is described in Algorithm 1 (insert-based local search) and Algorithm 2 (swap-based local search).

Algorithm 1. The insert-based local search.

```

1: Let  $\lambda$  denote a given batch sequence, where  $\lambda = \{b^1, b^2, \dots, b^l\}$  and  $l$  is the number of the batches, each of which contains
   at least one job;
2: Set  $k = 0$  and  $g = 0$ ;
3: while  $g < l$  do
4:   Set  $\lambda_{best} = null$  and  $C_{max}(\lambda_{best}) = +\infty$ ;
5:   Set  $k = k \% l + 1$  and  $k' = k$ ; // % denotes the modulus operator.
6:   while true do
7:     Remove the  $k$ th batch ( $b^k$ ) from  $\lambda$ . Insert  $b^k$  into the  $(k \% l + 1)$ th position of  $\lambda$ , resulting in a new schedule  $\lambda'$ ;
8:     Set  $k = k \% l + 1$ ;
9:     if  $k = k'$  then
10:       break;
11:     end if
12:     if  $C_{max}(\lambda') < C_{max}(\lambda_{best})$  then
13:       Update  $\lambda_{best} = \lambda'$ ;
14:     end if
15:   end while
16:   if  $C_{max}(\lambda_{best}) < C_{max}(\lambda)$  then
17:     Update  $g = 0$  and  $\lambda = \lambda_{best}$ ;
18:   else
19:     Set  $g = g + 1$ ;
20:   end if
21: end while

```

5.2 Framework of the EDA with local search

Based on the above sections, the general framework of the EDA with local search (EDA_{ls} for short) is summarised in Algorithm 3.

6. Computational experiments

6.1 Experimental design

To evaluate the proposed algorithm, problem instances were randomly generated in a way like Muthuswamy et al. (2012). Five factors that have effect on instances were identified: the number of jobs, the variation in job sizes, the variation in job processing times, the variation in job ready times and the set-up time on the second machine.

Random instances with 20, 50, 100, 200 and 300 jobs were generated. The job sizes s_j were generated from a discrete uniform distribution [1, 10] and [1, 20]. The job processing times p_{j1} and p_{j2} were generated from discrete uniform distributions [1, 50] and [1, 10], respectively. The job ready times r_j were generated from a uniform distribution [0, $R \sum_{j=1}^n p_{j1}$], where the factor R determines the relative frequency of job arrivals. Two set-up times T on the second machine were considered, 5 and 10. The capacity of the first machine S was assumed to be 20 for all instances. For each combination of factors and levels, 5 problem instances were randomly generated, thus, resulting in 200 instances. The experimental design is summarised in Table 4. Each category of problems is represented with a run code. For example, a problem category with 50 jobs, $T=10$, job sizes generated from [1, 10] and $R = 0.1$ is denoted by n2T2s1R1.

Algorithm 2. The swap-based local search.

```

1: Let  $\lambda$  denote a given batch sequence, where  $\lambda = \{b^1, b^2, \dots, b^l\}$  and  $l$  is the number of the batches, each of which contains
   at least one job;
2: Set  $a = 0$  and  $sum = l$ ;
3: while  $sum > 1$  do
4:   Set  $\lambda_{best} = null$  and  $C_{\max}(\lambda_{best}) = +\infty$ ;
5:   Set  $a = a \% l + 1$ ,  $sum = sum - 1$  and  $times = 0$ ;
6:   while  $times < sum$  do
7:     Set  $b = (a + times) \% l + 1$ ;
8:     Exchange the  $a$ th batch and the  $b$ th batch in  $\lambda$ , resulting in a new schedule  $\lambda'$ ;
9:     if  $C_{\max}(\lambda') < C_{\max}(\lambda_{best})$  then
10:      Update  $\lambda_{best} = \lambda'$ ;
11:   end if
12:   Exchange the  $a$ th batch and the  $b$ th batch back;
13:   Set  $times = times + 1$ ;
14: end while
15: if  $C_{\max}(\lambda_{best}) < C_{\max}(\lambda)$  then
16:   Update  $sum = l$  and  $\lambda = \lambda_{best}$ ;
17: end if
18: end while

```

Algorithm 3. The EDA_{IS} algorithm.

```

1: Initialize parameters  $Q$ ,  $\alpha$  and  $\beta$ ;
2: Create an initial population of individuals;
3: Set  $t = 0$ ;
4: while  $t < Maximum\_Generations$  do
5:   For each individual (corresponding to a job sequence), group jobs into batches using the BF heuristic, sequence the
   obtained batches by the LIWT heuristic and then calculate the  $C_{\max}$  value;
6:   Apply the insert-based local search and swap-based local search to the best solution orderly;
7:   Determine the best  $Q_1$  individuals from the current population;
8:   Update the probabilistic matrix with the selected individuals by Equation (17);
9:   Sample from the probabilistic matrix to generate new population;
10:  Update  $t = t + 1$ ;
11: end while
12: Output the best  $C_{\max}$  found so far;

```

Table 4. Summary of experimental design.

Factors	Levels	Number of levels
n	20, 50, 100, 200, 300	5
s_j	Uniform [1,10], uniform [1,20]	2
$p_{jm}, m = 1, 2$	$p_{j1} \sim \text{uniform}[1,50], p_{j2} \sim \text{uniform}[1,10]$	1
r_j	Uniform $[0, R \sum_{j=1}^n p_{j1}]$, $R = 0.1, 0.5$	2
T	5, 10	2
S	20	1
Instances per category		5
Total instances		200

Table 5. Simulation results under different combinations of α and β .

Run code	1	2	3	4	5	6	7	8	9	10	11	12	13
n3T1s1R1	985.4	985.4	985.4	983.4	983.2	985.4	985.4	985.4	985.0	982.5	985.4	985.4	985.4
n3T1s1R2	1533.4	1533.4	1533.4	1511.3	1525.2	1533.4	1533.4	1533.4	1523.7	1526.8	1533.4	1533.4	1533.4
n3T1s2R1	1470.6	1470.6	1468.5	1457.2	1461.4	1470.6	1470.6	1469.7	1463.2	1462.1	1470.6	1470.6	1470.0
n3T1s2R2	1793.2	1794.9	1772.2	1700.2	1729.8	1792.7	1792.0	1781.3	1732.2	1706.3	1798.0	1794.0	1785.0
n3T2s1R1	1117.8	1111.4	1099.0	1051.5	1072.5	1116.2	1115.5	1107.4	1063.7	1054.8	1114.4	1112.7	1110.1
n3T2s1R2	1645.3	1638.2	1609.8	1536.7	1582.8	1637.4	1646.6	1622.5	1563.8	1562.8	1642.2	1634.6	1635.9
n3T2s2R1	1630.0	1630.9	1605.2	1548.5	1575.6	1634.7	1628.8	1613.0	1573.2	1557.2	1634.0	1633.2	1617.5
n3T2s2R2	1856.6	1850.0	1818.2	1742.8	1787.0	1858.6	1854.4	1825.2	1769.5	1773.1	1863.5	1858.3	1835.2
Avg.	1504.0	1501.9	1486.5	1441.5	1464.7	1503.6	1503.3	1492.2	1459.3	1453.2	1505.2	1502.8	1496.6
	14	15	16	17	18	19	20	21	22	23	24	25	
n3T1s1R1	985.4	983.7	985.4	985.4	985.4	985.4	984.5	985.4	985.4	985.4	985.4	982.6	
n3T1s1R2	1531.8	1515.4	1533.4	1533.4	1533.4	1533.4	1514.3	1533.4	1533.4	1532.6	1533.4	1508.4	
n3T1s2R1	1467.8	1460.0	1470.6	1470.6	1470.6	1469.0	1460.6	1470.6	1470.6	1470.2	1469.6	1460.9	
n3T1s2R2	1763.9	1711.3	1797.0	1796.9	1789.5	1779.5	1705.6	1794.9	1790.8	1793.6	1782.9	1707.5	
n3T2s1R1	1079.8	1042.6	1113.8	1118.8	1110.8	1099.4	1039.4	1116.9	1114.8	1113.3	1102.4	1043.6	
n3T2s1R2	1586.4	1542.2	1635.8	1642.1	1632.6	1603.2	1542.3	1639.4	1631.9	1637.4	1619.8	1537.8	
n3T2s2R1	1593.9	1556.4	1634.1	1631.7	1617.4	1606.8	1552.0	1634.3	1632.5	1626.1	1613.8	1556.4	
n3T2s2R2	1805.0	1754.7	1861.7	1856.3	1841.2	1823.0	1751.4	1860.4	1855.5	1854.2	1838.4	1752.1	
Avg.	1476.8	1445.8	1504.0	1504.4	1497.6	1487.5	1443.8	1504.4	1501.9	1501.6	1493.2	1443.7	

Note: 1–25 represent the 25 combinations of (α, β) . For example, '2' represents combination (0.05, 0.01).

6.2 Parameter settings

A preliminary experiment was conducted to determine the parameters for the EDA. There are three parameters that affect solution quality and computational time obtained by the EDA. They are population size Q , parameter α associated with the best Q_1 individuals, and learning rate β . The population size was fixed to 40. In order to set α and β , different values of them were tested: $\alpha \in \{0.05, 0.1, 0.2, 0.3, 0.4\}$ and $\beta \in \{0.005, 0.01, 0.05, 0.1, 0.5\}$. This results in a total of 25 combinations. All the combinations were tested on the whole 100 job instances. For each instance, the EDA was run five independent times with a termination criterion of a maximum of 200 generations and the average makespan values were reported. The simulation results are shown in Table 5. The EDA was coded in JAVA. All the experiments were run on a Windows 7 Enterprise computer with AMD Dual Core processor 2.3 GHz and 4.0 GB of RAM.

Table 5 shows the fourth combination, i.e. (0.05, 0.1), achieves the best result among all the combinations. Thus, α and β are set to 0.05 and 0.1 in the further experiments.

6.3 Comparison of EDA and CPLEX

In this subsection, the mixed integer programming model given in Section 3 was solved using CPLEX and its results were compared with the EDA. CPLEX is a commercial solver for linear programming problems. In our experiments, we terminated

Table 6. Results of CPLEX and EDA.

Job number	CPLEX			EDA					Impr. (%) over CPLEX
	C_{\max}	GAP (%)	Run time	Best C_{\max}	Avg. C_{\max}	Worst C_{\max}	SD	Run time	
10	232	0.00	6.2	245	245.0	245	0.00	0.8	-5.60
15	397	27.67	1800.0	407	407.0	407	0.00	1.5	-2.52
20	332	18.37	1800.0	350	355.6	364	5.54	2.8	-5.42
25	585	56.37	1800.0	566	568.6	572	2.29	4.7	3.25
30	567	56.34	1800.0	563	567.2	572	3.43	7.4	0.71
35	620	54.19	1800.0	616	620.8	627	4.04	11.0	0.65
40	1156	70.85	1800.0	763	773.0	783	5.59	14.8	34.00
45	1344	70.46	1800.0	922	925.4	932	2.84	20.3	31.40
50	1386	70.60	1800.0	973	981.6	992	6.47	26.4	29.80
55	1349	65.42	1800.0	931	948.1	964	10.42	33.7	30.99
60	6252	92.59	1800.0	883	897.4	914	10.45	42.0	85.88
65	6828	94.58	1800.0	1041	1051.4	1066	8.36	52.3	84.75
70	7889	95.32	1800.0	1349	1373.7	1389	11.43	63.5	82.90
75	8472	94.56	1800.0	1222	1240.5	1259	11.23	75.6	85.58
80	8426	94.88	1800.0	1227	1254.3	1280	13.32	90.1	85.44
85	9912	95.21	1800.0	1592	1605.2	1632	10.58	106.5	83.94
90	—	—	1800.0	1508	1523.2	1539	9.88	125.1	—
95	10192	95.00	1800.0	1644	1667.4	1690	12.39	147.0	83.87
100	—	—	1800.0	1790	1809.4	1822	9.51	169.5	—

Note: '—' represents CPLEX cannot find a feasible solution in 1800 s.

CPLEX after allowing it to run for 1800 s and the best-known solution was used for comparison. Several random problem instances of category T1s2R2 were generated and the numbers of jobs varied from 10 to 100.

Table 6 shows the results obtained by CPLEX and the EDA for these instances. Column 1 represents the number of jobs for the instances. Columns 2, 3 and 4 report the C_{\max} , GAP and run time (s) obtained by CPLEX, respectively. Each instance was solved 10 times by the EDA. Columns 5, 6, 7 and 8 report the best, average, worst C_{\max} , and the standard deviation (SD) among the 10 runs of the EDA, respectively. Column 9 reports the run time (s) taken by the EDA. The last column reports the improvement realised through the EDA over CPLEX in terms of the solution, where $\text{Impr. (\%)} = 100 * [C_{\max}(\text{CPLEX}) - \text{Best } C_{\max}(\text{EDA})] / C_{\max}(\text{CPLEX})$.

As can be observed from Table 6, for the 10–20 job instances, the C_{\max} obtained by CPLEX is better than the best one obtained by the EDA. When the number of jobs is greater than or equal to 25, the EDA reports better results as compared to CPLEX. In addition, the EDA takes much less time than CPLEX.

6.4 Comparison of EDA and PSO

The EDA is compared with the PSO algorithm proposed by Muthuswamy et al. (2012). The parameters of the PSO algorithm were fixed the same as those in the literature except the number of particles and number of iterations. In our experiments, the two parameters were set to 100 particles and 1000 iterations for PSO.

Table 7 presents the results obtained by the EDA and PSO algorithms for 20 job instances. Column 1 represents the run code for the problem instances. Five instances were generated for each run code and each instance was solved five independent times by each algorithm. Columns 2, 3 and 4 report the average values of the best, average and worst C_{\max} of the five instances among the five runs of the EDA, respectively. Columns 5 and 6 report the standard deviation and the average run time (s) taken by the EDA for each instance. Columns 7–11 report those results obtained by the PSO algorithm. Column 12 reports the improvement realised through the EDA over PSO in terms of solution quality, where $\text{Impr. (\%)} \text{ over PSO} = 100 * [\text{Avg. } C_{\max}(\text{PSO}) - \text{Avg. } C_{\max}(\text{EDA})] / \text{Avg. } C_{\max}(\text{PSO})$. Tables 8–11 show the results obtained by the two algorithms for 50, 100, 200 and 300 job instances, respectively.

It can be seen from Table 7 that for all the 20 job instances, the solutions obtained by the EDA are better than those obtained by PSO. Moreover, the SD values of the solutions produced by the EDA are smaller when compared to PSO. It signifies that the EDA is more robust than PSO. On average, the EDA is better than PSO by 2.87%.

As can be observed from Tables 8–11, the solutions yielded by the EDA are better than the solutions yielded by PSO for all the categories of problem instances. Especially, the improvements realised through the EDA over PSO become more

Table 7. Results for 20 job instances.

Run code	EDA					PSO					Impr. (%) over PSO
	Best C_{\max}	Avg. C_{\max}	Worst C_{\max}	SD	Run time	Best C_{\max}	Avg. C_{\max}	Worst C_{\max}	SD	Run time	
n1T1s1R1	210.2	216.0	220.0	3.71	1.4	223.4	230.2	236.6	4.86	4.2	6.17
n1T1s1R2	304.4	309.1	313.0	3.02	1.4	307.0	313.8	322.8	5.64	3.8	1.50
n1T1s2R1	375.4	375.7	376.0	0.28	1.5	375.6	380.6	386.4	4.11	4.7	1.30
n1T1s2R2	352.2	356.4	362.6	3.78	1.3	359.4	366.5	376.0	6.00	4.4	2.76
n1T2s1R1	219.8	223.8	227.2	2.82	1.4	229.2	238.1	244.6	5.53	3.7	6.03
n1T2s1R2	312.6	315.8	319.2	2.65	1.3	314.2	322.1	329.0	5.14	3.7	1.95
n1T2s2R1	390.4	391.0	392.0	0.59	1.3	390.2	395.7	404.8	5.55	4.3	1.17
n1T2s2R2	370.4	373.1	378.4	2.95	1.4	376.0	380.9	386.4	4.10	4.2	2.04
Average	316.9	320.1	323.6	2.47	1.4	321.9	328.5	335.8	5.11	4.1	2.87

Table 8. Results for 50 job instances.

Run code	EDA					PSO					Impr. (%) over PSO
	Best C_{\max}	Avg. C_{\max}	Worst C_{\max}	SD	Run time	Best C_{\max}	Avg. C_{\max}	Worst C_{\max}	SD	Run time	
n2T1s1R1	485.0	496.7	506.4	8.25	12.5	534.8	545.2	556.0	8.48	12.2	8.91
n2T1s1R2	745.6	758.5	775.2	10.76	12.5	784.0	794.5	811.4	9.77	12.1	4.54
n2T1s2R1	860.0	864.4	871.4	4.20	12.7	926.6	944.4	962.0	12.64	15.1	8.47
n2T1s2R2	878.4	884.8	890.8	4.32	12.8	933.6	946.0	959.0	9.83	15.2	6.47
n2T2s1R1	506.2	516.7	525.2	7.39	12.3	555.4	567.2	579.2	8.73	11.9	8.90
n2T2s1R2	757.4	768.8	780.0	8.55	12.3	793.4	806.3	823.0	10.80	12.4	4.64
n2T2s2R1	880.0	886.8	891.8	4.23	12.6	940.4	957.4	977.8	12.78	14.7	7.38
n2T2s2R2	897.6	905.4	914.4	6.10	12.5	949.2	967.2	983.6	11.79	14.9	6.39
Average	751.3	760.3	769.4	6.73	12.5	802.2	816.0	831.5	10.60	13.6	6.96

Table 9. Results for 100 job instances.

Run code	EDA					PSO					Impr. (%) over PSO
	Best C_{\max}	Avg. C_{\max}	Worst C_{\max}	SD	Run time	Best C_{\max}	Avg. C_{\max}	Worst C_{\max}	SD	Run time	
n3T1s1R1	972.4	982.4	985.4	5.17	78.2	1077.4	1104.9	1137.8	22.14	36.2	11.09
n3T1s1R2	1486.0	1506.8	1517.6	11.45	77.8	1550.2	1580.0	1608.8	20.56	36.0	4.63
n3T1s2R1	1457.2	1458.9	1461.4	1.83	78.5	1645.4	1671.4	1700.4	21.44	46.4	12.72
n3T1s2R2	1685.2	1706.6	1722.8	14.16	78.9	1827.2	1858.6	1892.8	23.88	47.0	8.18
n3T2s1R1	1022.6	1032.7	1038.8	5.69	76.6	1111.0	1129.9	1148.6	13.78	35.6	8.61
n3T2s1R2	1509.6	1524.8	1537.8	10.56	76.4	1557.6	1589.0	1623.6	22.36	35.5	4.04
n3T2s2R1	1524.8	1529.3	1532.6	2.80	77.2	1710.0	1741.5	1767.0	19.55	46.4	12.18
n3T2s2R2	1725.4	1742.0	1763.2	13.02	77.5	1882.2	1902.5	1922.2	14.17	48.5	8.44
Average	1422.9	1435.4	1445.0	8.08	77.6	1545.1	1572.2	1600.2	19.74	41.5	8.73

pronounced as the number of jobs in the instances increases. For example, the EDA is better than PSO by, on average, 6.96% for 50 job instances, while the EDA is better than PSO by 14.01% for 300 job instances. Besides, for each problem category, the EDA achieves better the best, average and worst C_{\max} when compared to PSO. In addition, the SD values of the solutions produced by the EDA are consistently less than those produced by PSO. For example, the average SDs of the EDA and PSO algorithms are, respectively, 5.04 and 43.27 for 300 job instances, which indicates that the EDA has the better quality of convergence.

Figures 5–8 graphically show the effect of different problem factors, including number of jobs, job sizes, job ready times and setup times, on the average improvement in makespan when the EDA and PSO algorithms are compared, respectively. From these figures, it is evident that the EDA is more effective than PSO.

Table 10. Results for 200 job instances.

Run code	EDA					PSO					Impr. (%) over PSO
	Best C_{\max}	Avg. C_{\max}	Worst C_{\max}	SD	Run time	Best C_{\max}	Avg. C_{\max}	Worst C_{\max}	SD	Run time	
n4T1s1R1	1844.0	1844.0	1844.0	0.00	651.0	2167.0	2205.4	2239.8	25.28	119.9	16.39
n4T1s1R2	2784.8	2784.8	2784.8	0.00	637.9	3002.6	3046.3	3101.6	37.29	130.5	8.58
n4T1s2R1	2939.4	2939.4	2939.4	0.00	646.1	3442.4	3488.6	3535.8	36.96	163.6	15.74
n4T1s2R2	3415.2	3443.5	3478.0	22.04	637.0	3696.8	3742.0	3799.2	35.58	254.6	7.98
n4T2s1R1	1953.0	1953.0	1953.0	0.00	760.5	2206.2	2236.9	2269.8	22.25	122.4	12.69
n4T2s1R2	2789.8	2789.8	2789.8	0.00	656.5	3026.0	3080.8	3127.8	35.51	123.5	9.45
n4T2s2R1	3085.4	3085.4	3085.4	0.00	864.5	3579.2	3621.0	3665.4	31.67	167.0	14.79
n4T2s2R2	3499.4	3522.7	3552.8	18.51	633.0	3810.0	3844.4	3879.6	24.43	166.7	8.37
Average	2788.9	2795.3	2803.4	5.07	685.8	3116.3	3158.2	3202.4	31.12	156.0	11.75

Table 11. Results for 300 job instances.

Run code	EDA					PSO					Impr. (%) over PSO
	Best C_{\max}	Avg. C_{\max}	Worst C_{\max}	SD	Run time	Best C_{\max}	Avg. C_{\max}	Worst C_{\max}	SD	Run time	
n5T1s1R1	2827.0	2827.0	2827.0	0.00	2178.9	3384.0	3444.3	3499.4	40.01	249.1	17.92
n5T1s1R2	4163.4	4163.4	4163.4	0.00	2616.2	4729.4	4777.1	4829.6	35.91	325.5	12.85
n5T1s2R1	4551.0	4551.0	4551.0	0.00	2569.4	5487.2	5578.8	5642.0	59.02	434.6	18.42
n5T1s2R2	5159.0	5186.4	5208.6	19.15	2586.7	5618.4	5681.3	5727.4	37.86	578.0	8.71
n5T2s1R1	2992.4	2992.4	2992.4	0.00	2427.5	3440.8	3489.3	3527.4	29.39	253.0	14.24
n5T2s1R2	4175.0	4175.0	4175.0	0.00	2501.6	4734.2	4827.3	4902.0	62.00	254.2	13.51
n5T2s2R1	4742.4	4742.4	4742.4	0.00	2549.2	5670.2	5728.4	5790.8	40.84	363.1	17.21
n5T2s2R2	5245.6	5274.5	5302.0	21.20	2352.5	5746.0	5808.3	5862.2	41.14	350.0	9.19
Average	4232.0	4239.0	4245.2	5.04	2472.8	4851.3	4916.8	4972.6	43.27	350.9	14.01

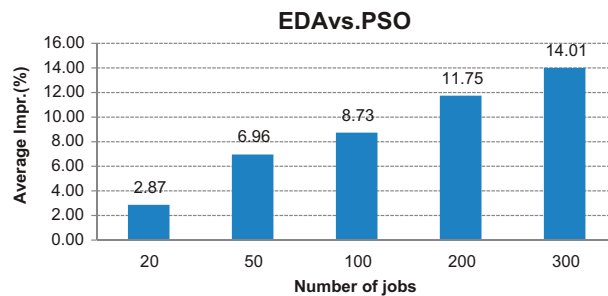


Figure 5. Effect of job number on makespan improvements.

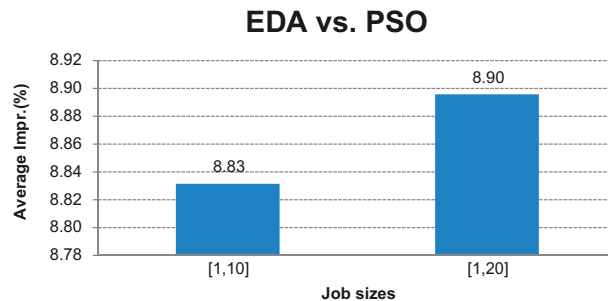


Figure 6. Effect of job size distribution on makespan improvements.

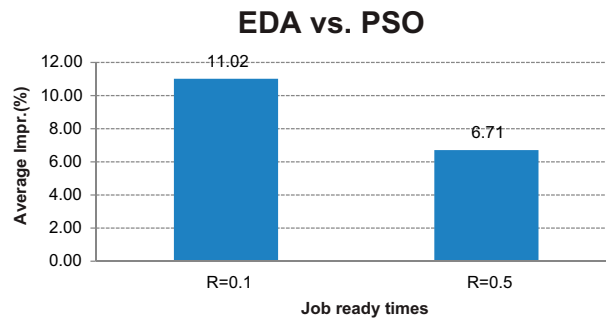


Figure 7. Effect of job ready time distribution on makespan improvements.

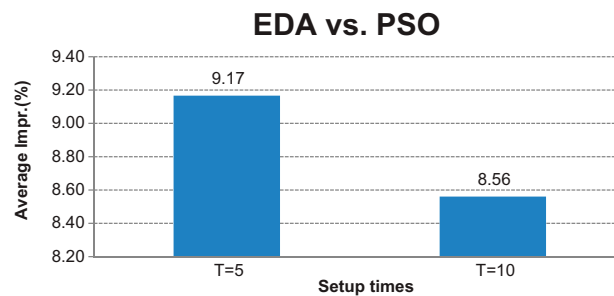


Figure 8. Effect of setup time on makespan improvements.

Table 12. Results of the EDA and the EDA_{ls} for 100 job instances.

Run code	EDA				EDA _{ls}				Impr. (%) over EDA
	Best C_{\max}	Avg. C_{\max}	SD	Run time	Best C_{\max}	Avg. C_{\max}	SD	Run time	
n3T1s1R1	972.4	982.4	5.17	78.2	951.6	953.5	0.96	78.8	2.94
n3T1s1R2	1486.0	1506.8	11.45	77.8	1473.8	1488.7	10.43	78.9	1.21
n3T1s2R1	1457.2	1458.9	1.83	78.5	1429.2	1432.2	2.88	84.2	1.83
n3T1s2R2	1685.2	1706.6	14.16	78.9	1651.0	1668.6	12.30	86.6	2.22
n3T2s1R1	1022.6	1032.7	5.69	76.6	1004.6	1013.2	4.84	82.7	1.88
n3T2s1R2	1509.6	1524.8	10.56	76.4	1495.2	1505.5	9.81	82.4	1.27
n3T2s2R1	1524.8	1529.3	2.80	77.2	1502.0	1506.2	3.03	88.7	1.51
n3T2s2R2	1725.4	1742.0	13.02	77.5	1685.6	1702.8	13.83	120.8	2.25
Average	1422.9	1435.4	8.08	77.6	1399.1	1408.9	7.26	87.9	1.89

With respect to the run times of the EDA and PSO algorithms, experimental results show the run time of the EDA is less than that of PSO for smaller scale problems (20 and 50 jobs). The EDA requires more run time compared to PSO as the number of jobs increases. This is due to the fact that the EDA generates new population by sampling from the probabilistic model.

6.5 Comparison of EDA and EDA_{ls}

In order to see the effectiveness of combining the EDA algorithm and problem-dependent local search, computational experiments and comparison were conducted between the EDA and the EDA_{ls} algorithms. The computational results are given in Tables 12–14.

It is easily observed from Tables 12–14 that the solutions obtained by the EDA_{ls} are better than those obtained by the EDA. On average, the improvement of the EDA_{ls} over the EDA in terms of solution quality is 1.89% for 100 job instances, 1.55% for 200 job instances, and 1.33% for 300 job instances, respectively. This indicates the fact that incorporating the local

Table 13. Results of the EDA and the EDA_{IS} for 200 job instances.

Run code	EDA				EDA _{IS}				Impr. (%) over EDA
	Best C_{\max}	Avg. C_{\max}	SD	Run time	Best C_{\max}	Avg. C_{\max}	SD	Run time	
n4T1s1R1	1844.0	1844.0	0.00	651.0	1800.6	1800.6	0.00	829.3	2.35
n4T1s1R2	2784.8	2784.8	0.00	637.9	2762.4	2762.4	0.00	689.2	0.80
n4T1s2R1	2939.4	2939.4	0.00	646.1	2890.4	2890.4	0.00	823.5	1.67
n4T1s2R2	3415.2	3443.5	22.04	637.0	3317.0	3359.2	29.11	810.1	2.45
n4T2s1R1	1953.0	1953.0	0.00	760.5	1929.6	1929.6	0.00	838.4	1.20
n4T2s1R2	2789.8	2789.8	0.00	656.5	2771.4	2771.4	0.00	665.8	0.66
n4T2s2R1	3085.4	3085.4	0.00	864.5	3039.0	3039.6	0.32	956.2	1.48
n4T2s2R2	3499.4	3522.7	18.51	633.0	3431.8	3459.6	18.67	729.0	1.79
Average	2788.9	2795.3	5.07	685.8	2742.8	2751.6	6.01	792.7	1.55

Table 14. Results of the EDA and the EDA_{IS} for 300 job instances.

Run code	EDA				EDA _{IS}				Impr. (%) over EDA
	Best C_{\max}	Avg. C_{\max}	SD	Run time	Best C_{\max}	Avg. C_{\max}	SD	Run time	
n5T1s1R1	2827.0	2827.0	0.00	2178.9	2776.8	2776.8	0.00	2694.3	1.78
n5T1s1R2	4163.4	4163.4	0.00	2616.2	4140.4	4140.4	0.00	2929.8	0.55
n5T1s2R1	4551.0	4551.0	0.00	2569.4	4496.0	4496.0	0.00	2671.3	1.21
n5T1s2R2	5159.0	5186.4	19.15	2586.7	5034.8	5058.1	18.42	2656.8	2.47
n5T2s1R1	2992.4	2992.4	0.00	2427.5	2962.2	2962.2	0.00	2916.6	1.01
n5T2s1R2	4175.0	4175.0	0.00	2501.6	4148.4	4148.4	0.00	2535.1	0.64
n5T2s2R1	4742.4	4742.4	0.00	2549.2	4689.6	4689.6	0.00	3143.3	1.11
n5T2s2R2	5245.6	5274.5	21.20	2352.5	5149.6	5177.5	18.67	3362.7	1.84
Average	4232.0	4239.0	5.04	2472.8	4174.7	4181.1	4.64	2863.7	1.33

search in the EDA algorithm leads to further improvement in solution quality. Compared to the EDA, the EDA_{IS} requires longer run time. However, its solution quality is better.

7. Conclusions

This paper investigated the problem of minimising makespan in a no-wait flowshop with two batch processing machines (BPMs), non-identical job sizes and arbitrary ready times. A parallel BPM (p-batch) is followed by a serial BPM (s-batch). The problem was formulated as a mixed integer linear programming model. We then proposed a population-based evolutionary method named estimation of distribution algorithm (EDA), in which individuals were represented as job sequences. A probabilistic model was built to generate new individuals and an incremental learning mechanism was developed to update the probabilistic model. The best-fit heuristic was adopted to form batches and a least idle/waiting time approach was proposed to schedule the batches. Two local search heuristics were embedded in the EDA to further enhance the exploitation ability. Computational experiments and comparisons demonstrated the superiority of the proposed algorithm for solving the problem under study in terms of solution quality and robustness. The effectiveness of the local search was evaluated.

There are a few important directions for future research. First of all, this research can be extended to optimise other objectives such as total completion time, completion time variance and so on. Adding due date constraints to the jobs is also of interest as it may arise in real-world applications. In addition, this approach could be extended to flowshop problems with more than two stages. Finally, BPM problems with incompatible job families can be considered as well.

Funding

This work was supported by the National Natural Science Foundation of China [grant number 71171184]. This research also received funding from the University of Tennessee Health Information Technology & Simulation (HITS) Laboratory.

Disclosure statement

No potential conflict of interest was reported by the authors.

References

- Abedi, Mehdi, Hany Seidgar, Hamed Fazlollahtabar, and Rohollah Bijani. 2015. "Bi-objective Optimisation for Scheduling the Identical Parallel Batch-processing Machines with Arbitrary Job Sizes, Unequal Job Release Times and Capacity Limits." *International Journal of Production Research* 53 (6): 1680–1711.
- Ahmadi, J. H., R. H. Ahmadi, S. Dasu, and C. S. Tang. 1992. "Batching and Scheduling Jobs on Batch and Discrete Processors." *Operations Research* 40 (4): 750–763.
- Aickelin, Uwe, Edmund K. Burke, and Jingpeng Li. 2006. "An Estimation of Distribution Algorithm with Intelligent Local Search for Rule-based Nurse Rostering." *Journal of the Operational Research Society* 58 (12): 1574–1585.
- Baluja, Shumeet, and Scott Davies. 1997. "Using Optimal Dependency-trees for Combinatorial Optimization: Learning the Structure of the Search Space." Technical Report. CMU-CS-97-107. Pittsburgh, PA: Carnegie Mellon University.
- Baluja, Shumeet, and Scott Davies. 1998. "Fast Probabilistic Modeling for Combinatorial Optimization." In *AAAI/IAAI*, Madison, WI, USA, 469–476.
- Behnamian, J., S. M. T. Fatemi Ghomi, F. Jolai, and O. Amirtaheri. 2012. "Realistic Two-stage Flowshop Batch Scheduling Problems with Transportation Capacity and Times." *Applied Mathematical Modelling* 36 (2): 723–735.
- Bramel, Julien, and David Simchi-Levi. 1998. *The Logic of Logistics: Theory, Algorithms and Applications for Logistics Management*. Berlin: Springer.
- Chang, Pei-Chann, Shih-Hsin Chen, Chin-Yuan Fan, and V. Mani. 2010. "Generating Artificial Chromosomes with Probability Control in Genetic Algorithm for Machine Scheduling Problems." *Annals of Operations Research* 180 (1): 197–211.
- Chang, Pei-Chann, Jih-Chang Hsieh, Shih-Hsin Chen, Jun-Lin Lin, and Wei-Hsiu Huang. 2009. "Artificial Chromosomes Embedded in Genetic Algorithm for a Chip Resistor Scheduling Problem in Minimizing the Makespan." *Expert Systems with Applications* 36 (3): 7135–7141.
- Chen, Huaping, Du Bing, and George Q. Huang. 2010. "Metaheuristics to Minimise Makespan on Parallel Batch Processing Machines with Dynamic Job Arrivals." *International Journal of Computer Integrated Manufacturing* 23 (10): 942–956.
- Chen, Huaping, Shengchao Zhou, Xueping Li, and Xu Rui. 2014. "A Hybrid Differential Evolution Algorithm for a Two-stage Flow Shop on Batch Processing Machines with Arbitrary Release Times and Blocking." *International Journal of Production Research* 52 (19): 5714–5734.
- Chen, Shih-Hsin, Pei-Chann Chang, T. C. E. Cheng, and Qingfu Zhang. 2012. "A Self-guided Genetic Algorithm for Permutation Flowshop Scheduling Problems." *Computers & Operations Research* 39 (7): 1450–1457.
- Chen, Yuh-Min, Min-Chih Chen, Pei-Chann Chang, and Shih-Hsin Chen. 2012. "Extended Artificial Chromosomes Genetic Algorithm for Permutation Flowshop Scheduling Problems." *Computers & Industrial Engineering* 62 (2): 536–545.
- Cheng, Bayi, Shanlin Yang, Hu Xiaoxuan, and Kai Li. 2014. "Scheduling Algorithm for Flow Shop with Two Batch-processing Machines and Arbitrary Job Sizes." *International Journal of Systems Science* 45 (3): 571–578.
- Cheng, T. C. E., B. M. T. Lin, and A. Toker. 2000. "Makespan Minimization in the Two-machine Flowshop Batch Scheduling Problem." *Naval Research Logistics* 47 (2): 128–144.
- Cheng, T. C. E., and G. Q. Wang. 1998. "Batching and Scheduling to Minimize the Makespan in the Two-machine Flowshop." *IIE Transactions* 30 (5): 447–453.
- Damodaran, Purushothaman, Praveen Kumar Manjeshwar, and Krishnaswami Srihari. 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., and K. Srihari. 2004. "Mixed Integer Formulation to Minimize Makespan in a Flow Shop with Batch Processing Machines." *Mathematical and Computer Modelling* 40 (13): 1465–1472.
- De Bonet, Jeremy S., Ch L. Isbell, and Paul Viola. 1997. "MIMIC: Finding Optima by Estimating Probability Densities." In *Advances in Neural Information Processing Systems*, edited by Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, 424–430. Cambridge: MIT Press.
- Ghazvini, Fariborz Jolai, and Lionel Dupont. 1998. "Minimizing Mean Flow Times Criteria on a Single Batch Processing Machine with Non-identical Jobs Sizes." *International Journal of Production Economics* 55 (3): 273–280.
- Glass, C. A., C. N. Potts, and V. A. Strusevich. 2001. "Scheduling Batches with Sequential Job Processing for Two-machine Flow and Open Shops." *INFORMS Journal on Computing* 13 (2): 120–137.
- Gong, Hua, Lixin Tang, and C. W. Duin. 2010. "A Two-stage Flow Shop Scheduling Problem on a Batching Machine and a Discrete Machine with Blocking and Shared Setup Times." *Computers & Operations Research* 37 (5): 960–969.
- Graham, Ronald L., Eugene L. Lawler, Jan Karel Lenstra, and A. H. G. Kan. 1979. "Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey." *Annals of Discrete Mathematics* 5 (2): 287–326.
- Harik, Georges. 1999. "Linkage Learning via Probabilistic Modeling in the ECGA." (IlligAL Report NO. 99010). Illinois: Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign.
- Harik, Georges R., Fernando G. Lobo, and David E. Goldberg. 1999. "The Compact Genetic Algorithm." *IEEE Transactions on Evolutionary Computation* 3 (4): 287–297.

- Jia, Wenyong, Zhibin Jiang, and You Li. 2015. "Combined Scheduling Algorithm for Re-entrant Batch-processing Machines in Semiconductor Wafer Manufacturing." *International Journal of Production Research* 53 (6): 1866–1879.
- Jolai, F., H. Kor, S. M. Hatefi, and H. Iranmanesh. 2009. "A Genetic Algorithm for Makespan Minimization in a No-wait Flow Shop Problem with Two Batching Machines." In *International Conference on Computer Engineering and Technology*. Vol. 2238–2242. Los Alamitos, CA: IEEE.
- Kashan, A. H., and B. Karimi. 2009. "An Improved Mixed Integer Linear Formulation and Lower Bounds for Minimizing Makespan on a Flow Shop with Batch Processing Machines." *International Journal of Advanced Manufacturing Technology* 40 (5–6): 582–594.
- Lee, Chung-Yee, Reha Uzsoy, and Louis A. Martin-Vega. 1992. "Efficient Algorithms for Scheduling Semiconductor Burn-in Operations." *Operations Research* 40 (4): 764–775.
- Li, Xuesong, Hiroaki Ishii, and Minghao Chen. 2015. "Single Machine Parallel-batching Scheduling Problem with Fuzzy Due-date and Fuzzy Precedence Relation." *International Journal of Production Research* 53 (9): 2707–2717.
- Liao, Ching-Jong, and Li-Man Liao. 2008. "Improved MILP Models for Two-machine Flowshop with Batch Processing Machines." *Mathematical and Computer Modelling* 48 (7): 1254–1264.
- Liao, Li-Man, and Ching-Jen Huang. 2011. "Tabu Search Heuristic for Two-machine Flowshop with Batch Processing Machines." *Computers & Industrial Engineering* 60 (3): 426–432.
- Lin, B. M. T., and T. C. E. Cheng. 2001. "Batch Scheduling in the No-wait Two-machine Flowshop to Minimize the Makespan." *Computers & Operations Research* 28 (7): 613–624.
- Lin, B. M. T., and T. C. E. Cheng. 2005. "Two-machine Flowshop Batching and Scheduling." *Annals of Operations Research* 133 (1–4): 149–161.
- Lin, Rock, and Ching-Jong Liao. 2013. "Batch Scheduling Problem for a Machinery Factory with Fixed-position Layout." *International Journal of Production Research* 51 (3): 910–926.
- Liu, Bojun, Yushun Fan, and Yi Liu. 2015. "A Fast Estimation of Distribution Algorithm for Dynamic Fuzzy Flexible Job-shop Scheduling Problem." *Computers & Industrial Engineering* 87 (C): 193–201.
- Lozano, Jose A., Pedro Larranaga, Inaki Inza, and Endika Bengoetxea. 2006. *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms*. Vol. 192. Berlin: Springer.
- Manjeshwar, P. K., P. Damodaran, and K. Srihari. 2009. "Minimizing Makespan in a Flow Shop with Two Batch-processing Machines Using Simulated Annealing." *Robotics and Computer-Integrated Manufacturing* 25 (3): 667–679.
- Manjeshwar, Praveen Kumar, Purushothaman Damodaran, and Krishnaswami Srihari. 2011. "Genetic Algorithms for Minimizing Makespan in a Flow Shop with Two Capacitated Batch Processing Machines." *International Journal of Advanced Manufacturing Technology* 55 (9–12): 1171–1182.
- Mathirajan, M., and A. I. Sivakumar. 2006. "Minimizing Total Weighted Tardiness on Heterogeneous Batch Processing Machines with Incompatible Job Families." *International Journal of Advanced Manufacturing Technology* 28 (9–10): 1038–1047.
- Mirsanei, H. S., B. Karimi, and F. Jolai. 2009. "Flow Shop Scheduling with Two Batch Processing Machines and Nonidentical Job Sizes." *International Journal of Advanced Manufacturing Technology* 45 (5–6): 553–572.
- Muhlenbein, Heinz, and Thilo Mahnig. 1999. "Convergence Theory and Applications of the Factorized Distribution Algorithm." *Journal of Computing and Information Theory* 7 (1): 19–32.
- Muhlenbein, H., and G. Paass. 1996. "From Recombination of Genes to the Estimation of Distributions I: Binary Parameters." *Lecture Notes in Computer Science* 1141: 178–187.
- Muthuswamy, Shanthi, Mario C. Velez-Gallego, Jairo Maya, and Miguel Rojas-Santiago. 2012. "Minimizing Makespan in a Two-machine No-wait Flow Shop with Batch Processing Machines." *International Journal of Advanced Manufacturing Technology* 63 (1–4): 281–290.
- Oulamara, A. 2007. "Makespan Minimization in a No-wait Flow Shop Problem with Two Batching Machines." *Computers & Operations Research* 34 (4): 1033–1050.
- Oulamara, Ammar, Gerd Finke, and A. Kamgaing Kuiteing. 2009. "Flowshop Scheduling Problem with a Batching Machine and Task Compatibilities." *Computers & Operations Research* 36 (2): 391–401.
- Oulamara, Ammar, Mikhail Y. Kovalyov, and Gerd Finke. 2005. "Scheduling a No-wait Flow Shop Containing Unbounded Batching Machines." *IIE Transactions* 37 (8): 685–696.
- Pan, Quan-Ke, and Rubén Ruiz. 2012. "An Estimation of Distribution Algorithm for Lot-streaming Flow Shop Problems with Setup Times." *Omega* 40 (2): 166–180.
- Pedro, Larraaga, and Jose A. Lozano. 2002. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Vol. 2. Norwell, MA: Kluwer Academic Publishers.
- Pelikan, Martin. 2005. *Hierarchical Bayesian Optimization Algorithm: Toward a New Generation of Evolutionary Algorithms*, edited by Janusz Kacprzyk, 31–48. Berlin Heidelberg: Springer.
- Pelikan, Martin, David E. Goldberg, and Fernando G. Lobo. 2002. "A Survey of Optimization by Building and Using Probabilistic Models." *Computational Optimization and Applications* 21 (1): 5–20.
- Pelikan, Martin, and Heinz Mühlenbein. 1999. "The Bivariate Marginal Distribution Algorithm." In *Advances in Soft Computing*, edited by Rajkumar Roy, Takeshi Furuhashi, and Pravir K. Chawdhry, 521–535. London: Springer.
- Potts, C. N., V. A. Strusevich, and T. Tautenhahn. 2001. "Scheduling Batches with Simultaneous Job Processing for Two-machine Shop Problems." *Journal of Scheduling* 4 (1): 25–51.

- Rossi, Andrea, Andrea Pandolfi, and Michele Lanzetta. 2014. "Dynamic Set-up Rules for Hybrid Flow Shop Scheduling with Parallel Batching Machines." *International Journal of Production Research* 52 (13): 3842–3857.
- Sagarna, Ramón, and Jose Antonio Lozano. 2005. "On the Performance of Estimation of Distribution Algorithms Applied to Software Testing." *Applied Artificial Intelligence* 19 (5): 457–489.
- Su, L. H. 2003. "A Hybrid Two-stage Flowshop with Limited Waiting Time Constraints." *Computers & Industrial Engineering* 44 (3): 409–424.
- Sung, C. S., and Y. H. Kim. 2002. "Minimizing Makespan in a Two-machine Flowshop with Dynamic Arrivals Allowed." *Computers & Operations Research* 29 (3): 275–294.
- Sung, C. S., and Y. H. Kim. 2003. "Minimizing Due Date Related Performance Measures on Two Batch Processing Machines." *European Journal of Operational Research* 147 (3): 644–656.
- Sung, C. S., Y. H. Kim, and S. H. Yoon. 2000. "A Problem Reduction and Decomposition Approach for Scheduling for a Flowshop of Batch Processing Machines." *European Journal of Operational Research* 121 (1): 179–192.
- Sung, C. S., and J. I. Min. 2001. "Scheduling in a Two-Machine Flowshop with Batch Processing Machine(s) for Earliness/Tardiness Measure Under a Common Due Date." *European Journal of Operational Research* 131 (1): 95–106.
- Tang, Lixin, Hua Gong, Jiyin Liu, and Feng Li. 2014. "Bicriteria Scheduling on a Single Batching Machine with Job Transportation and Deterioration Considerations." *Naval Research Logistics* 61 (4): 269–285.
- Tang, Lixin, and Peng Liu. 2009a. "Flowshop Scheduling Problems with Transportation or Deterioration between the Batching and Single Machines." *Computers & Industrial Engineering* 56 (4): 1289–1295.
- Tang, Lixin, and Peng Liu. 2009b. "Minimizing Makespan in a Two-machine Flowshop Scheduling with Batching and Release Time." *Mathematical and Computer Modelling* 49 (5–6): 1071–1077.
- Wang, Ling, Shengyao Wang, Xu Ye, Gang Zhou, and Min Liu. 2012. "A Bi-population Based Estimation of Distribution Algorithm for the Flexible Job-shop Scheduling Problem." *Computers & Industrial Engineering* 62 (4): 917–926.
- Wang, Sheng-yao, Ling Wang, Min Liu, and Xu Ye. 2013. "An Effective Estimation of Distribution Algorithm for Solving the Distributed Permutation Flow-shop Scheduling Problem." *International Journal of Production Economics* 145 (1): 387–396.
- Zhang, Yi, and Xiaoping Li. 2011. "Estimation of Distribution Algorithm for Permutation Flow Shops with Total Flowtime Minimization." *Computers & Industrial Engineering* 60 (4): 706–718.
- Zhou, Shengchao, Huaping Chen, Xu Rui, and Xueping Li. 2013. "Minimising Makespan on a Single Batch Processing Machine with Dynamic Job Arrivals and Non-identical Job Sizes." *International Journal of Production Research* 52 (8): 2258–2274.