Omega 40 (2012) 166–180

Contents lists available at ScienceDirect

Omega

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

An estimation of distribution algorithm for lot-streaming flow shop problems with setup times

Quan-Ke Pan [a], Rubén Ruiz [b,*]

[a] College of Computer Science, Liaocheng University, Liaocheng 252059, PR China
[b] Grupo de Sistemas de Optimización Aplicada, Instituto Tecnológico de Informática, Ciudad Politécnica de la Innovación, Edifico 8G, Acc. B. Universitat Politécnica de Valéncia, Camino de Vera s/n, 46022 Valencia, Spain

ARTICLE INFO

Article history:
Received 7 July 2010
Accepted 9 May 2011
Processed by Associate editor Zhang
Available online 12 May 2011

Keywords:
Flow shop scheduling
Lot-streaming
Estimation of distribution algorithm
Makespan
Sequence-dependent setup times

ABSTRACT

Lot-streaming flow shops have important applications in different industries including textile, plastic, chemical, semiconductor and many others. This paper considers an n-job m-machine lot-streaming flow shop scheduling problem with sequence-dependent setup times under both the idling and no-idling production cases. The objective is to minimize the maximum completion time or makespan. To solve this important practical problem, a novel estimation of distribution algorithm (EDA) is proposed with a job permutation based representation. In the proposed EDA, an efficient initialization scheme based on the NEH heuristic is presented to construct an initial population with a certain level of quality and diversity. An estimation of a probabilistic model is constructed to direct the algorithm search towards good solutions by taking into account both job permutation and similar blocks of jobs. A simple but effective local search is added to enhance the intensification capability. A diversity controlling mechanism is applied to maintain the diversity of the population. In addition, a speed-up method is presented to reduce the computational effort needed for the local search technique and the NEH-based heuristics. A comparative evaluation is carried out with the best performing algorithms from the literature. The results show that the proposed EDA is very effective in comparison after comprehensive computational and statistical analyses.

1. Introduction

The permutation flow shop scheduling problem is one of the most extensively studied combinatorial optimization problems. It has important applications, among others, in manufacturing systems, assembly lines and information service facilities in use nowadays. In a traditional flow shop, there are n jobs that have to be processed on m machines. All jobs visit the machines in the same sequence. Each job is assumed to be indivisible, and thus, it cannot be transferred to the downstream machine until the whole operation on the preceding machine is finished. Nevertheless, this is not the case in many practical environments where a job or lot consists of many identical items. For example, in the fastener production process, jobs are batches of thousands of bolts, dowels, or rivets. The whole batch does not need to be finished in order to move on to the next machine. Another example comes from the electronics and semiconductor production environment where a job comprises thousands of identical electronic components and again it is not necessary to wait for all items in the lot to be completed before moving to the downstream machine. In order to accelerate production, a job is allowed to overlap its operations between successive machines by splitting it into a number of smaller sub-lots and moving the completed portion of the sub-lots to downstream machines [62]. More examples arise in the ceramic tile sector where batches of ceramic tiles are composed of literally thousands of ceramic tiles. When going from the molding and glaze decoration lines to the kiln firing machines, the whole batch of tiles does not need to be fully completed and overlapping is desirable in practice. The process of splitting jobs into sub-lots is usually called lot-streaming, which was first introduced by Reiter [40] and has become one of the most effective techniques used to implement time-based strategies in today's global competition [8,50]. Generally, there are two different production situations when processing the sub-lots of a job, namely, the idling case and no-idling case. In the no-idling case, jobs must be continuously processed without interruptions (i.e., idle time) between any two adjacent sub-lots of the same job. The idling case allows idle time on machines. It is known that makespan based on the idling case is shorter than that based on the no-idling case under the same sub-lot type [8]. However, both cases have their respective practical applications in today's competitive production environments. With regards to the potential benefits of lot streaming, they are mentioned by Truscott [55] as follows: (a) reduction in production lead times (thus, leading to better due-date performance); (b) reduction in work-in-process inventory and associated costs; (c) reductions in interim storage and

* Corresponding author. Tel.: +34 96 387 70 07x74946; fax: +34 96 387 74 99.
E-mail addresses: panquanke@gmail.com (Q.-K. Pan), rruiz@eio.upv.es (R. Ruiz).

0305-0483/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.omega.2011.05.002

space requirements; and (d) reduction in material handling system capacity requirements. Therefore, in recent years, lot streaming has received extensive attention and has been applied to flow shop scheduling problems starting with the work of Tseng and Liao [56].

Setup times involve non-productive operations such as cleaning, obtaining or adjusting tools, fixing or releasing parts to machines and others. Setup times are very important in practice as noted in Allahverdi and Soroush [3]. Although they are not part of the job processing times, these operations have to be done prior to the processing of the jobs. Setup times can be broadly classified in two categories [1,2]. The first category is referred to as sequence-independent setup, where setups depend only on the machine and on the next job to be processed. The second one is sequence-dependent setup, in which setups depend not only on the job to be processed next but also on its immediately preceding job on the same machine. An example is given by Ruiz and Allahverdi [43]: in the painting industry, after producing a black paint, substantial cleaning must be performed if one intends to produce a white paint, while less cleaning is necessary if a batch of dark gray paint is to be produced. On the other hand, almost no cleaning is required when production is changed from a sub-lot of the black paint to another one of a similar black paint.

This paper considers lot-streaming flow shop scheduling problems with sequence-dependent setup times, with important applications, as commented, in textile, plastic, chemical and semiconductor industries. Without loss of generality, this problem is denoted as $F_m,L_n|prmu,ST_{sd}|C_{\max}$ by using the well known $\alpha/\beta/\gamma$ notation with the extensions of Chang and Chiu [8] and Allahverdi et al. [1], where ST_{sd} represents the sequence-dependent setup time and F_m and L_n stand for the n-job m-machine lot-streaming flow shop configuration. The permutation flow shop scheduling problem under makespan criterion is already NP-Hard as was shown by Garey et al. [12] (for three or more machines, i.e., $m \geq 3$). Since we consider lot-streaming and sequence-dependent setup times, the studied problem is also NP-Hard.

Estimation of distribution algorithms (EDA) were introduced by Mühlenbein and Paass [31]. EDA are a class of novel population-based evolutionary algorithms. Unlike traditional evolutionary algorithms, EDA samples new solutions from a probabilistic model which characterizes the distribution of promising solutions in the search space at each generation. Due to its effectiveness and search ability, EDA has recently attracted much attention in the field of evolutionary computation [21], and it has already been applied to solve combinatorial optimization problems, including the flow shop scheduling problem in Jarboui et al. [15] or more complex hybrid flow shop settings in Salhi et al. [48]. Therefore, EDA seems like a promising venue of research for the studied scheduling problem. However, to the best of our knowledge, there is no published work dealing with the lot-streaming version of flow shop scheduling problem using EDA, let alone with sequence-dependent setup times. In this paper we study this important and practical $F_m,L_n|prmu,ST_{sd}|\ C_{\max}$ problem. Furthermore, both the no-idling and idling cases are considered. The proposed EDA makes extensive use of some effective techniques like a NEH-based initialization, a sequence-representation-based probabilistic model, diversity measures and an insert-neighborhood-based local search. Computational experiments and statistical comparisons show that the proposed EDA outperforms the best performing algorithms that have recently appeared for solving the lot-streaming flow shop scheduling problem.

The rest of the paper is organized as follows: in Section 2, the literature on the lot-streaming flow shop scheduling problem is reviewed. In Section 3, the lot-streaming flow shop scheduling problem with sequence-dependent setup times is stated and formulated. Section 4 gives a brief introduction to the basic EDA methodology and presents our proposed EDA method in detail. Section 5 contains the calibration of the proposed EDA. The computational results and comparisons are provided in Section 6. Finally, concluding remarks are presented in Section 7.

2. Literature review

Having so many practical applications, lot-streaming has been extensively studied in the academic as well as in the industrial fields since the late 1980s [8]. Some papers deal with single-job lot-streaming problems, where the main goal is to determine the best allocation of sub-lots or the size of each sub-lot so as to minimize some given performance measures.

There are some important theoretical or basic results to highlight. First, Potts and Baker [37] indicated that it was sufficient to use the same sub-lot sizes for all machines as regards makespan criterion. This is an important result for the flow shop problem as different number of sub-lots for the machines would complicate the problems significantly. However, it remains to be seen if this result holds when sequence-dependent setup times are present. Furthermore, Baker and Jia [4] showed that makespan improved with the number of sub-lots. While this is an expected result (the more sub-lots the higher the machine utilization), the paper of Baker and Jia [4] actually quantifies and deeply studies the effect. Lastly, Trietsch and Baker [54] generalized some important structural properties by reviewing the different forms of single-job lot-streaming in the literature.

Apart from these theoretical results, many papers have been published where different lot-streaming flow shop settings and objectives are studied. Many of them are now discussed in chronological order. Kropp and Smunt [19] presented optimal sub-lot size policies and two heuristic methods for flow time minimization in a flow shop setting with no additional constraints. Vickson and Alfredsson [60] studied the effect of batch transfer in a two-machine and special three-machine flow shop problems with unit-size sub-lots. Çetinkaya [7] proposed an optimal transfer batch and scheduling algorithm for a two-stage problem with separated setup, processing and removal times. Vickson [59] examined a two-machine problem involving setup and sub-lot transfer times with respect to both continuous and integer valued sub-lot sizes and some exact algorithms were presented. Sriskandarajah and Wagneur [51] presented an efficient heuristic for solving the problem of simultaneous lot-streaming and scheduling of multiple products in a two-machine no-wait flow shop. For the m-machine lot-streaming flow shop problem, Kumar et al. [20] extended the approach presented by Sriskandarajah and Wagneur [51] to the m-machine case. Later, Kalir and Sarin [16] proposed a bottleneck minimal idleness heuristic to sequence a set of batches to be processed in equal sub-lots for minimizing makespan. Yoon and Ventura [63] developed sixteen pairwise interchange methods to optimize the mean weighted absolute deviation from due dates. To the best of our knowledge, this is the first study about lot-streaming flow shop involving due dates. Bukchin et al. [6] examined the optimal solution properties and developed a solution procedure for a two-machine flow shop scheduling problem with sub-lot detached setups and batch availability. Liu [23] proposed a heuristic method for discrete lot streaming with variable sub-lots in a flow shop. Kalir and Sarin [17] developed a near optimal solution procedure for the determination of the number of sub-lots as well as the sequence in a flow shop lot streaming problem with sub-lot-attached setups.

Zhang et al. [64] developed two heuristic algorithms for the multi-job lot-streaming problem in a two-stage hybrid flow shop with the objective to minimize the mean completion time of the jobs. Marimuthu and Ponnambalam [26] proposed a genetic algorithm (GA) and a simulated annealing (SA) for lot streaming in a two-machine flow shop to minimize makespan. Liu et al. [24] studied the multi-product variable lot streaming in a flow shop.

A hybrid heuristic was applied for determining product sequences, lot streaming reallocation machines, and lot streaming ranges by combining a tabu search (TS) with simulated annealing (SA). Additionally, a linear programming model was used to find the minimal makespan and lot streaming for each machine and each product. Feldmann and Biskup [10] provided a mixed integer programming formulation for the multi-product lot streaming problem in a permutation flow shop with intermingling of sub-lots from different jobs. While in this paper we do not consider intermingling, (where not all sub-lots of the same job follow one another in a sequence), it is a very promising venue of research.

Recently, more complex single-job lot-streaming problems were addressed. Liu [22] investigated the continuous version of the problem and provided optimal and heuristic solution methods for the general problem. Edis and Ornek [9] proposed a heuristic by combining simulation and tabu search to minimize the makespan for a single-product multistage stochastic flow shop problem with consistent sub-lot types and discrete sub-lot sizes. Kim and Jeong [18] proposed a self-adaptive genetic algorithm for scheduling a flow shop problem with no-wait flexible lot-streaming constraints, where the splitting of order quantities of different products into sub-lots and alternative machines with different processing times was dealt with. Martin [30] presented a hybrid genetic algorithm/mathematical programming approach for a multi-family flow shop scheduling problem with lot streaming.

Most of the literature studies the lot streaming flow shop scheduling problem with fixed sizes of sub-lots under the non-intermingled case. For example, Yoon and Ventura [63] presented a hybrid genetic algorithm (HGA) to minimize the mean weighted absolute deviation of job completion times from their due dates. Tseng and Liao [56] developed a discrete particle swarm optimization (DPSO) algorithm. It was shown by the authors that their DPSO algorithm performed much better than the HGA proposed by Yoon and Ventura [63] for solving 900 randomly generated instances. More recently, Pan et al. [33] presented a discrete artificial bee colony (DABC) algorithm which outperformed both previous DPSO and HGA algorithms. Marimuthu et al. [27,28,29] applied a tabu search (TS), simulated annealing (SA), hybrid genetic algorithm (HGA), ant colony optimization (ACO) and threshold accepting (TA) algorithms, respectively, to deal with both makespan and total flow time criteria for a flow shop problem involving setup times. For multiobjective problems, Huang and Yang [14] presented a scheduling mechanism and an ant colony optimization heuristic for an overlap manufacturing problem with various ready times and sequence-dependent setup times.

As we can see from the previous review, and to the best of our knowledge, no metaheuristic has been applied to minimize the makespan in the n-job m-machine lot-streaming flow shop problem with sequence-dependent setup times. A comprehensive review of scheduling problems involving lot-streaming can be found in Chang and Chiu [8] and in Sarin and Jaiprakash [50].

3. Lot-streaming flow shop scheduling problem

This paper considers an n-job m-machine lot-streaming flow shop scheduling problem. The statement of the problem and an illustrative example are described in this section.

3.1. Statement of the problem

We assume that each job j can be split into a number $l(j)$ of smaller sub-lots with equal size such that $l(j)$ is the same for all machines. This follows the research work of Yoon and Ventura [62], Yoon and Ventura [63], Tseng and Liao [56], Marimuthu et al. [27,28,29]. Once the processing of a sub-lot on its preceding machine is completed, it can be transferred to the downstream machine immediately. However, all $l(j)$ sub-lots of job j should be processed continuously as no intermingling is allowed. A separable sequence-dependent setup time is necessary for the first sub-lot of each job j before it can be processed on any machine k. Furthermore, at any time, each machine can process at most one sub-lot and each sub-lot can be processed on at most one machine. Let the processing time of each sub-lot of job j on machine k be $p(k,j)$, and the setup time of job j on machine k, after having processed job j' is $s(k,j',j)$. For simplicity, let $s(k,j,j)$ represent the setup time of job j if it is the first job to be proceeded in the machine. The objective is to find a sequence with the optimal sub-lot starting and completion times to minimize the makespan.

Let a job permutation $\pi=\{\pi_1,\pi_2,\ldots,\pi_n\}$ represent the sequence of jobs to be processed, and $ST(k,j,e)$ and $CT(k,j,e)$ denote the earliest start time and the earliest completion time of the eth sub-lot of job j on machine k, respectively. $C_{\max}(\pi)$ denotes the makespan of the schedule under permutation π. Then, for the idling case, $C_{\max}(\pi)$ can be calculated as follows:

$$\begin{cases} ST(1,\pi_1,1)=s(1,\pi_1,\pi_1) \\ CT(1,\pi_1,1)=ST(1,\pi_1,1)+p(1,\pi_1) \\ ST(k,\pi_1,1)=\max\{CT(k-1,\pi_1,1),s(k,\pi_1,\pi_1)\}, \quad k=2,3,\ldots,m \\ CT(k,\pi_1,1)=ST(k,\pi_1,1)+p(k,\pi_1), \quad k=2,3,\ldots,m \end{cases} \tag{1}$$

$$\begin{cases} ST(1,\pi_1,e)=CT(1,\pi_1,e-1), \quad e=2,3,\ldots,l(\pi_1) \\ CT(1,\pi_1,e)=ST(1,\pi_1,e)+p(1,\pi_1), \quad e=2,3,\ldots,l(\pi_1) \\ ST(k,\pi_1,e)=\max\{CT(k-1,\pi_1,e),CT(k,\pi_1,e-1)\}, \quad e=2,3,\ldots,l(\pi_1), \ k=2,3,\ldots,m \\ CT(k,\pi_1,e)=ST(k,\pi_1,e)+p(k,\pi_1), \quad e=2,3,\ldots,l(\pi), \ k=2,3,\ldots,m \end{cases} \tag{2}$$

$$\begin{cases} ST(1,\pi_i,1)=CT(1,\pi_{i-1},l(\pi_{i-1}))+s(1,\pi_{i-1},\pi_i), \quad i=2,3,\ldots,n \\ CT(1,\pi_i,1)=ST(1,\pi_i,1)+p(1,\pi_i), \quad i=2,3,\ldots,n \\ ST(k,\pi_i,1)=\max\{CT(k-1,\pi_i,1),CT(k,\pi_{i-1},l(\pi_{i-1}))+s(k,\pi_{i-1},\pi_i)\}, \\ i=2,3,\ldots,n, \ k=2,3,\ldots,m \\ CT(k,\pi_i,1)=ST(k,\pi_i,1)+p(k,\pi_i), i=2,3,\ldots,n, \ k=2,3,\ldots,m \end{cases} \tag{3}$$

$$\begin{cases} ST(1,\pi_i,e)=CT(1,\pi_i,e-1), \quad i=2,3,\ldots,n, \ e=2,3,\ldots,l(\pi_i) \\ CT(1,\pi_i,e)=ST(1,\pi_i,e)+p(1,\pi_i), \quad i=2,3,\ldots,n, \ e=2,3,\ldots,l(\pi_i) \\ ST(k,\pi_i,e)=\max\{CT(k-1,\pi_i,e),CT(k,\pi_i,e-1)\}, \\ i=2,3,\ldots,n, \ e=2,3,\ldots,l(\pi_i), \ k=2,3,\ldots,m \\ CT(k,\pi_i,e)=ST(k,\pi_i,e)+p(k,\pi_i), \quad i=2,3,\ldots,n, \ e=2,3,\ldots,l(\pi_i), \ k=2,3,\ldots,m \end{cases} \tag{4}$$

$$C_{\max}(\pi)=CT(m,\pi_n,l(\pi_n)) \tag{5}$$

Correspondingly, $C_{\max}(\pi)$for the no-idling case is calculated as follows:

$$\begin{cases} ST(1,\pi_1,1)=s(1,\pi_1,\pi_1) \\ CT(1,\pi_1,l(\pi_1))=ST(1,\pi_1,1)+l(\pi_1)\times p(1,\pi_1) \end{cases} \tag{6}$$

$$\begin{cases} \quad ST(k,\pi_1,1)=\max\{s(k,\pi_1,\pi_1),ST(k-1,\pi_1,1)+p(k-1,\pi_1), \\ \quad CT(k-1,\pi_1,l(\pi_1))-(l(\pi_1)-1)\times p(k,\pi_1)\}, \quad k=2,3,\ldots,m \\ CT(k,\pi_1,l(\pi_1))=ST(k,\pi_1,1)+l(\pi_1)\times p(k,\pi_1), \quad k=2,3,\ldots,m \end{cases} \tag{7}$$

$$\begin{cases} ST(1,\pi_i,1)=CT(1,\pi_{i-1},l(\pi_{i-1}))+s(1,\pi_{i-1},\pi_i), \quad i=2,3,\ldots,n \\ CT(1,\pi_i,l(\pi_i))=ST(1,\pi_i,1)+l(\pi_i)\times p(1,\pi_i), \quad i=2,3,\ldots,n \end{cases} \tag{8}$$

$$\begin{cases} \qquad ST(k,\pi_i,1)=\max\{ST(k-1,\pi_i,1)+p(k-1,\pi_i), \\ \quad CT(k-1,\pi_i,l(\pi_i))-(l(\pi_i)-1)\times p(k,\pi_i), \\ \quad CT(k,\pi_{i-1},l(\pi_{i-1}))+s(k,\pi_{i-1},\pi_i)\}, \quad i=2,3,\ldots,n; \ k=2,3,\ldots,m \\ CT(k,\pi_i,l(\pi_i))=ST(k,\pi_i,1)+l(\pi_i)\times p(k,\pi_i), \quad i=2,3,\ldots,n; \ k=2,3,\ldots,m \end{cases} \tag{9}$$

$$C_{\max}(\pi) = CT(m,\pi_n,l(\pi_n)) \tag{10}$$

Then the objective of the lot streaming flow shop scheduling problem with makespan criterion is to find a permutation π^* in the set of all permutations Π such that

$$C_{\max}(\pi^*) \le C_{\max}(\pi), \quad \forall \pi \in \Pi \tag{11}$$

In equation set (1), the first and third equations specify the earliest start time for the first sub-lot of job π_1, where both the completion time of the sub-lot on the previous machine and the setup time of the job on the current machine are considered. The second and fourth equations calculate the completion times, making sure that preemption of jobs is not allowed. Equation set (2) controls the earliest start time and the earliest completion time for the successive sub-lots of job π_1, which ensure that sub-lots of the same job are processed continuously. Equation sets (3) and (4) contain the calculations for the sub-lots of the following jobs in the sequence. When calculating the start time for the first sub-lot of a job in set (3), we take into account the completion time of the previous job on the current machine, the completion time of the sub-lot on the previous machine, and the setup time of the job on the current machine. Finally, from Eq. (5), we can see that the makespan is equivalent to the completion time of the last sub-lot of the last job π_n on the last machine.

Equation sets (6)–(10) consider the makespan for the no-idling case. In sets (6) and (7), the top equations give the earliest start time for the first sub-lot of job π_1. We can see that the earliest start time is equal to the maximum value among the setup time of the job on the current machine, the completion time of the first sub-lot on the previous machine, and the difference between the completion time of the whole job on the previous machine and the total processing time of the whole job on the preceding machine except the last sub-lot, which ensures that no idling interruption time exists between any two adjacent sub-lots of the same job. The bottom equations calculate the earliest completion time for the last sub-lot of job π_1. Sets (8) and (9) control the calculation of the subsequent jobs in the permutation. Different from sets (6) and (7), we need consider the completion time of the last sub-lot of the previous job on the preceding machine when calculating the earliest start time.

3.2. Illustrative example

The following example illustrates the calculation for a four-job, three-machine instance with a given permutation $\pi=\{1,2,3,4\}$. Let us give the necessary data for the example:

$[l(j)]_{1\times 4}=[2,2,1,2]$, i.e., jobs 1, 2 and 4 contain two sub-lots and job 3, just one sub-lot

$$[p(k,j)]_{3\times 4} = \begin{bmatrix} 4 & 3 & 2 & 5 \\ 2 & 2 & 2 & 3 \\ 2 & 2 & 3 & 5 \end{bmatrix}$$

$$[s(k,j',j)]_{3\times 4\times 4} = \left\{ \begin{bmatrix} 2 & 1 & 2 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & 2 \\ 2 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 1 & 2 & 1 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 2 & 2 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 2 \\ 1 & 2 & 2 & 1 \end{bmatrix} \right\}$$

For the idling case, the makespan is calculated below and the Gantt chart is shown in Fig. 1

$$ST(1,1,1) = s(1,1,1) = 2$$
$$CT(1,1,1) = ST(1,1,1) + p(1,1) = 2+4 = 6$$

$$ST(2,1,1) = \max\{CT(1,1,1), s(2,1,1)\} = \max\{6,2\} = 6$$
$$CT(2,1,1) = ST(2,1,1) + p(2,1) = 6+2 = 8$$

$$ST(3,1,1) = \max\{CT(2,1,1), s(3,1,1)\} = \max\{8,2\} = 8$$
$$CT(3,1,1) = ST(3,1,1) + p(3,1) = 8+2 = 10$$

$$ST(1,1,2) = CT(1,1,1) = 6$$
$$CT(1,1,2) = ST(1,1,2) + p(1,1) = 6+4 = 10$$

$$ST(2,1,2) = \max\{CT(1,1,2), CT(2,1,1)\} = \max\{10,8\} = 10$$
$$CT(2,1,2) = ST(2,1,2) + p(2,1) = 10+2 = 12$$

$$ST(3,1,2) = \max\{CT(2,1,2), CT(3,1,1)\} = \max\{12,10\} = 12$$
$$CT(3,1,2) = ST(3,1,2) + p(3,1) = 12+2 = 14$$

$$ST(1,2,1) = CT(1,1,2) + s(1,1,2) = 10+1 = 11$$
$$CT(1,2,1) = ST(1,2,1) + p(1,2) = 11+3 = 14$$

$$ST(2,2,1) = \max\{CT(1,2,1), CT(2,1,2) + s(2,1,2)\} = \max\{14, 12+1\} = 14$$
$$CT(2,2,1) = ST(2,2,1) + p(2,2) = 14+2 = 16$$

$$ST(3,2,1) = \max\{CT(2,2,1), CT(3,1,2) + s(3,1,2)\} = \max\{16, 14+1\} = 16$$
$$CT(3,2,1) = ST(3,2,1) + p(3,1) = 16+2 = 18 \text{ and so on until } C_{\max} = CT(3,4,2) = 40$$

For the no-idling case, the makespan is calculated below and the Gantt chart is shown in Fig. 2

$$ST(1,1,1) = s(1,1,1) = 2$$
$$CT(1,1,2) = ST(1,1,1) + l(1) \times p(1,1) = 2 + 2 \times 4 = 10$$

$$ST(2,1,1) = \max\{s(2,1,1), ST(1,1,1) + p(1,1), CT(1,1,2) - (l(1)-1) \times p(2,1)\}$$
$$= \max\{2, 2+4, 10-1 \times 2\} = 8$$

$$CT(2,1,2) = ST(2,1,1) + l(1) \times p(2,1) = 8 + 2 \times 2 = 12$$

$$ST(3,1,1) = \max\{s(3,1,1), ST(2,1,1) + p(2,1), CT(2,1,2) - (l(1)-1) \times p(3,1)\}$$
$$= \max\{2, 8+2, 12-1 \times 2\} = 10$$

$$CT(3,1,2) = ST(3,1,1) + l(1) \times p(3,1) = 10 + 2 \times 2 = 14$$

$$ST(1,2,1) = CT(1,1,2) + s(1,1,2) = 10+1 = 11$$
$$CT(1,2,2) = ST(1,2,1) + l(2) \times p(1,2) = 11 + 2 \times 3 = 17$$

$$ST(2,2,1) = \max\{ST(1,2,1) + p(1,2), CT(1,2,2) - (l(2)-1) \times p(2,2), CT(2,1,2) + s(2,1,2)\}$$
$$= \max\{11+3, 17-1 \times 2, 12+1\} = 15$$

$$CT(2,2,2) = ST(2,2,1) + l(2) \times p(2,2) = 15 + 2 \times 2 = 19$$

$$ST(3,2,1) = \max\{ST(2,2,1) + p(2,2), CT(2,2,2) - (l(2)-1) \times p(3,2), CT(3,1,2) + s(3,1,2)\}$$
$$= \max\{15+2, 19-1 \times 2, 14+1\} = 17$$

$$CT(3,2,2) = ST(3,2,1) + l(2) \times p(3,2) = 17 + 2 \times 2 = 21,$$
$$\text{and so on until } C_{\max} = CT(3,4,2) = 42$$

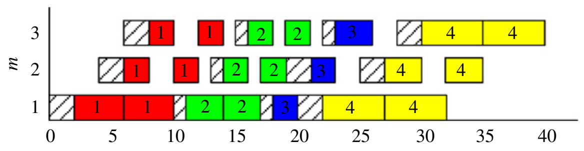

Fig. 1. Gantt chart for the idling case example.

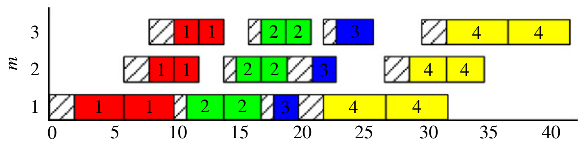

Fig. 2. Gantt chart for the no-idling case example.

4. Proposed EDA for the lot-streaming flow shop problem

EDA is a new metaheuristic methodology proposed by Mühlenbein and Paass [31], which is based on populations that evolve within the search process and has a theoretical foundation in probability theory. Instead of using the conventional crossover and mutation operations of regular genetic algorithms, EDA adopts a probabilistic model learned from a population of selected individuals to produce new solutions at each generation. Starting from a population of *PS* randomly generated individuals, EDA estimates a probabilistic model from the information of the selected *Q* individuals in the current generation, and represents it by conditional probability distributions for each decision variable. *M* offspring are then sampled in the search space according to the estimated probabilistic model. Finally, the next population is determined by replacing some individuals in the current generation with new generated offspring. The above steps are repeated until some stopping criterion is reached. The pseudo code for the basic EDA is summarized as follows [21]:

```
begin
    Generate a population of PS individuals randomly;
    Calculate fitness for each individual;
    while termination criterion not met, do
        Select Q individuals and estimate a probabilistic model;
        Sample M offspring from the estimated probabilistic model;
        Evaluate the M generated offspring;
        Generate new population;
    end while
end
```

We now detail the proposed EDA for solving the lot-streaming flow shop scheduling problem involving sequence-dependent setup times to minimize makespan. We explain the solution representation, population initialization, probabilistic model, generation of new individuals, population update, local search procedure and a diversity controlling mechanism in the next sections.

4.1. Solution representation and population initialization

One of the key issues when designing EDA lies in the solution representation where individuals bear the necessary information related to the problem domain at hand. The permutation based representation indicates the job processing order by machines. This representation has been widely used in the literature for a variety of permutation flow shop scheduling problems [45,58,15]. Therefore, we also employ it in this study.

The EDA method is formed by a population of *PS* individuals or *n*-job permutations. To guarantee an initial population with a certain level of quality and diversity, a common trend is to construct a few good individuals by effective heuristics and to produce others randomly. This initialization approach ensures a faster convergence to good solutions, and it is widely used in evolutionary algorithms developed for traditional flow shop scheduling problems [58]. It has been long known that the NEH heuristic [32] is a high performer for flow shop scheduling problems under different scenarios [11,44,38]. In this research, we extend it to handle the studied problem, and obtain two heuristics, referred to as NEH1 and NEH2, respectively. The NEH1 is obtained by modifying the objective evaluation in the basic NEH heuristic with the calculations described in Section 3. NEH1 can be described as follows:

Step 1: An initial permutation $\pi=\{\pi_1,\pi_2,\ldots,\pi_n\}$ is generated by sorting jobs in decreasing sum of their total processing times, i.e., $\Sigma_{k=1}^{m} p(k,j)\times l(j)$, $j=1,2,\ldots,n$.

Step 2: A job permutation is established by evaluating the partial sequences based on the obtained initial order. Suppose a current sequence $\pi'=\{\pi'_1,\pi'_2,\ldots,\pi'_i\}$ is already determined for the first i jobs of the initial permutation π, then $i+1$ partial sequences are constructed by inserting job π_{i+1} into the $i+1$ possible positions of the current sequence. Among these $i+1$ partial sequences, the one with the minimum makespan is kept as the current sequence for the next iteration. This step is repeated by considering job π_{i+2} and so on until all the jobs have been scheduled.

NEH2 has the same steps as NEH1 with the exception that the step 1 is modified as explained below:

Step 1: An initial permutation $\pi=\{\pi_1,\pi_2,\ldots,\pi_n\}$ is generated by sorting jobs in decreasing sum of their total processing times and mean setup times, i.e., $\Sigma_{k=1}^{m}((p(k,j)\times l(j)+\Sigma_{j'=1}^{n} s(k,j',j))/n)$, $j=1, 2, \ldots, n$.

There are a total of $(n-1)(n+2)/2$ partial sequences generated in step 2, so the computational complexity is $O(mn^3)$ in both no-idling and idling cases using the calculations presented in Section 3. For the basic NEH heuristic, a speed-up method was proposed by Taillard [53] resulting in an improved complexity of $O(mn^2)$. Later, the method was extended to the permutation flow shop problem with setup times [41], no-wait flow shop problem [34,36], no-idle flow shop problem [35], blocking flow shop problem [61] and others. Accelerations are very effective for flow shop problems. Rad et al. [38] stated that a very efficient NEH implementation with accelerations results in CPU times of only 77 ms for instances as large as 500×20 on a PIV 3.2 GHz PC computer. Non-accelerated versions can take up to 30 s for the same problem size. Therefore, we propose makespan calculation accelerations for the lot-streaming flow shop problem with setup times, which results in NEH1 and NEH2 to have a computational complexity of just $O(mn^2)$. This acceleration is now explained below:

Let $STb(k,j,e)$ be the latest start time of the eth sub-lot of job j on machine k in the backward pass calculation, that is, we proceed from the end of the sequence to the beginning. The procedure to evaluate the $i+1$ partial sequences when inserting job π_{i+1} into the $i+1$ possible positions of the partial permutation $\pi'=\{\pi'_1,\pi'_2,\ldots,\pi'_i\}$ can be simplified in the following way:

Step 1: Get $CT(k,\pi'_z,l(\pi'_z))$ for $z=1, 2, \ldots, i$ and $k=1, 2, \ldots, m$.
Step 2: Get $STb(k,\pi'_z,1)$ for $z=i, i-2, \ldots, 1$ and $k=m-1, m-2, \ldots, 1$.
Step 3: Repeat the following steps until all possible positions q, $q=\{1,2,\ldots,i+1\}$, of the permutation $\pi'=\{\pi'_1,\pi'_2,\ldots,\pi'_i\}$ are calculated:
Step 3.1: Insert job π_{i+1} into position q and generate a partial permutation π''.
Step 3.2: Calculate $CT(k,\pi''_q,l(\pi''_q))$ by using the previously calculated $CT(k,\pi'_{q-1},l(\pi'_{q-1}))$, where $k=1, 2, \ldots, m$. Note that $\pi''_q=\pi_{i+1}$.
Step 3.3: The makespan of the permutation π'' is given as follows (see in Figs. 3 and 4):

$$C_{\max}(\pi'')=\max_{k=1}^{m}(CT(k,\pi''_q,l(\pi''_q))+s(k,\pi''_q,\pi'_q)+STb(k,\pi'_q,1))$$

Clearly, both NEH1 and NEH2 heuristics result in a computational complexity of $O(mn^2)$ by using the above procedure to evaluate the generated partial sequences. With the presented

NEH1 and NEH2, we propose a population initialization procedure with both a high quality and a high diversity as follows:

Step 1: Generate an individual using NEH1.
Step 2: Generate an individual using NEH2. If it is different from the individual generated by NEH1, put it into population; otherwise discard it.
Step 3: Randomly produce an individual in the solution space. If it is different from all existing individuals, put it into the population; otherwise discard it. Repeat step 3 until the population has *PS* individuals.

The *PS* individuals of the population are always stored in ascending order of their makespan values.

4.2. Selection operator and probabilistic model

The probabilistic model construction represents the main part of the EDA method, which is estimated from the genetic information of the individuals chosen from the population by a selection operator. In classic evolutionary algorithms, roulette and tournament selection operators are commonly used. Such selection operators either require fitness and a mapping calculation or the individuals to be continuously compared and sorted. In this paper, we select the *Q* best individuals from the population to estimate a probabilistic model. Since individuals are stored in ascending order of their makespan values, we can complete the operator by selecting the first *Q* individuals in the population. This results in a very fast selection operator.

The performance of the EDA is closely related to the probabilistic model, and obviously, a good model can enhance the algorithm's efficiency and effectiveness for optimizing the problem

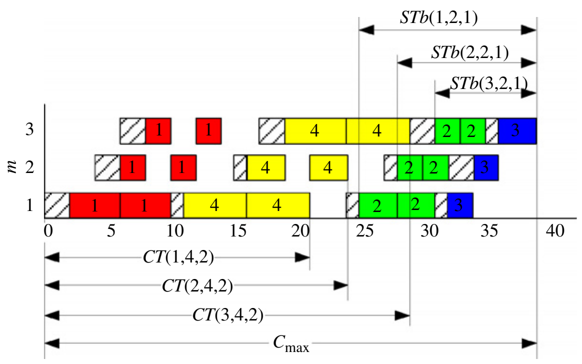

Fig. 3. Insert job '4' into the second position of the permutation $\pi=\{1,2,3\}$. Idling case.

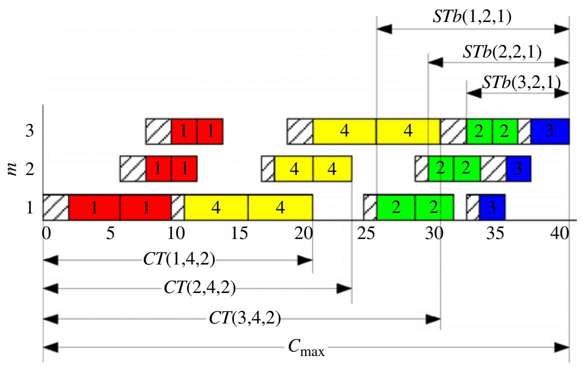

Fig. 4. Insert job '4' into the second position of the permutation $\pi=\{1,2,3\}$. No-idling case.

considered. Thus, the best choice of the model is crucial for designing an effective EDA. For solving the permutation flow shop scheduling problem with total flow time criterion, Jarboui et al. [15] presented a probabilistic model based on both the order of the jobs in the sequence and on similar blocks of jobs present in the selected individuals, which is described as follows:

Let $\rho_{i,j}$ be the number of times that job *j* appears before or in position *i* in the selected individuals, and $\tau_{j',j}(i)$ the number of times that job *j* appears immediately after job *j'* when job *j'* is in position $i-1$. Then, $\eta_{i,j}=\delta_1\times\rho_{i,j}$ and $\mu_{j',j}(i)=\delta_2\times\tau_{j',j}(i)$ indicate the importance of the order of jobs and of the similar blocks of jobs in the selected sequences, respectively, where δ_1 and δ_2 are two parameters used for the diversification of the solutions. Then, the probability for positioning job *j* in the *i*th position of the offspring is determined by

$$\xi_{i,j}=\frac{\eta_{i,j}\times\mu_{j',j}(i)}{\Sigma_{l\in\Omega(i)}\eta_{i,l}\times\mu_{j',l}(i)} \tag{12}$$

where $\Omega(i)$ is the set of jobs not scheduled until position *i* and *j'* is the job in the $(i-1)$th position of the offspring.

There are some shortcomings in the EDA model presented by Jarboui et al. [15]. First, as shown in Ruiz et al. [45], there are many similar blocks of jobs within the individuals' sequences in the latter stages of evolutionary methods. If these blocks are disrupted, the algorithm has a high probability to produce offspring with worse makespan values. These similar blocks may occupy the same positions or different positions. However, only the blocks in the same positions are considered by Jarboui et al. [15]. Second, according to the definition of $\tau_{j',j}(i)$, it is equal to zero when $i=1$, since job *j* is the first job in the sequence and no job *j'* is located before it. This results in the probability of selection of any job *j* in the first position to be always equal to zero. In other words, the first job of the offspring is determined randomly and not according to genetic information. Finally, if at an early stage of the algorithm there are not enough blocks in the same position, and $\tau_{j',j}(i)$ is equal to zero for most of jobs, only a few jobs with $\tau_{j',j}(i)>0$ are selected for producing offspring. Thus, the population easily looses diversity. To address the above shortcomings, we present a new probabilistic model, which is now detailed:

Let $\lambda_{j',j}$ represent the number of times that job *j* appears immediately after job *j'* in the selected individuals, which indicates the importance of similar blocks of jobs not only in the same positions but also in different positions as well. Then, the probability of placing job *j* in the *i*th position of the offspring is given by

$$\xi_{i,j}=\begin{cases}\dfrac{\rho_{i,j}}{\Sigma_{l\in\Omega(i)}\rho_{i,l}} & i=1\\[2ex] \dfrac{((\rho_{i,j})/(\Sigma_{l\in\Omega(i)}\rho_{i,l})+(\lambda_{j',j})/(\Sigma_{l\in\Omega(i)}\lambda_{j',l}))}{2} & i=2,3,\ldots,n\end{cases} \tag{13}$$

An example with four jobs is used to illustrate the presented probabilistic model. Suppose the selected individuals are $\pi(1)=\{1,2,3,4\}$, $\pi(2)=\{2,3,4,1\}$ and $\pi(3)=\{1,4,2,3\}$. Therefore, $\rho_{i,j}$ and $\lambda_{j',j}$ are given below

$$[\rho_{i,j}]_{4\times4}=\begin{bmatrix}2&1&0&0\\2&2&1&1\\2&3&2&2\\3&3&3&3\end{bmatrix},\quad [\lambda_{j',j}]_{4\times4}=\begin{bmatrix}-&1&0&1\\0&-&3&0\\0&0&-&2\\1&1&0&-\end{bmatrix},$$

Then, we calculate the probability of selection of each job in $\Omega(1)=\{1,2,3,4\}$ for the first position as follows: $\xi_{1,1}=2/(2+1)=0.67$, $\xi_{1,2}=1/(2+1)=0.33$, $\xi_{1,3}=0/(2+1)=0$, $\xi_{1,4}=0/(2+1)=0$.

Suppose job 1 was selected for the first position and $\Omega(2)=\{2,3,4\}$, then we calculate the probability of section of each job in $\Omega(2)=\{2,3,4\}$ as follows:

$$\xi_{2,2}=(2/(2+1+1)+1/(1+0+1))/2=0.5$$
$$\xi_{2,3}=(1/(2+1+1)+0/(1+0+1))/2=0.125$$
$$\xi_{2,4}=(1/(2+1+1)+1/(1+0+1))/2=0.375$$

4.3. Generation of new individuals and population update

Inspired by the algorithm developed by Rajendran and Ziegler [39] and the DPSO algorithm by Tseng and Liao [56],we present a procedure to generate a new sequence $\pi'=\{\pi'_1,\pi'_2,\ldots,\pi'_n\}$. Starting from an empty sequence, the procedure constructs π' by choosing a job for the first position, followed by choice of the second job, and so on. The pseudo code of the constructing procedure is given as follows:

```
begin
    for i=1 to n do
        if rand () < ε then
            Choose the first unscheduled job in the reference
            sequence.
        else
            Select job j according to probability $\xi_{i,j}$.
        endif
    endfor
end
```

In the above procedure, ε is a control parameter; *rand* () is a function returning a random number sampled from a uniform distribution between 0 and 1. The reference sequence is randomly chosen from the selected individuals for estimating the probabilistic model. When *rand* () $\geq \varepsilon$, we randomly select θ jobs from the unscheduled job set and the job with the largest $\xi_{i,j}$ is put into the ith position of the new sequence π'. To generate M offspring, the above procedure is repeated M times so to sample M offspring from the probabilistic model.

Another aspect considered in the EDA is the population update for the next generation. To maintain the diversity of the population and to avoid cycling the search, the population is updated in the following way [45]:

Step 1: Set $i=1$.
Step 2: If offspring i is better than the worst individual of the population and if there is no other identical individual in the population, replace the worst individual by i, otherwise, discard i.
Step 3: Set $i=i+1$, if $i\leq M$, go to step 2; otherwise stop the procedure.

4.4. Local search

It is natural to add a local search into the EDA to carry out intensification. We employ a local search based on the job insertion operator, which is very suitable for performing a fine local search and that is commonly used to produce a neighboring solution in the flow shop literature [46,58]. In this local search, a job is extracted from its original position in the sequence and reinserted in all other $n-1$possible positions. If a better makespan value is found, the solution is replaced. We repeat the procedure until no improvements are found. According to the extraction order of jobs in the first step, the local search can be classified as referenced local search [34] and local search without order [46]. Let $\pi^b=\{\pi^b_1,\pi^b_2,\ldots,\pi^b_n\}$ denote the best job sequence found so far, and $\pi=\{\pi_1,\pi_2,\ldots,\pi_n\}$ be a sequence that undergoes local search. Then the referenced local search is described as follows:

Step 1: Set $i=1$ and a counter *Cnt* to 0.
Step 2: Find job π^b_i in permutation π and record its position.
Step 3: Take out job π^b_i from its original position in π. Then insert it in another different position of π , and adjust the permutation accordingly by not changing the relative positions of the other jobs. Consider all the possible insertion positions and denote the best obtained sequence as π^*.
Step 4: If π^* is better than π, then set $\pi=\pi^*$ and $Cnt=0$; otherwise set $Cnt=Cnt+1$.
Step 5: If $Cnt<n$, let $i=\begin{cases} i+1 & i<n \\ 1 & i=n \end{cases}$, and go to step 2, otherwise output the current permutation π and stop.

The local search without order is sensibly different:

Step 1: Set counter $Cnt=0$.
Step 2: Remove a job at random from its original position in π without repetition. Then insert it in another different position of π , and adjust the permutation accordingly by not changing the relative positions of the other jobs. Consider all the possible insertion positions and denote the best obtained sequence as π^*.
Step 3: If π^* is better than π, then let $\pi=\pi^*$.
Step 4: Let $Cnt=Cnt+1$. If $Cnt<n$, go to step 2.
Step 5: If the permutation π was improved in the above steps 1 through 4, then go to step 1; otherwise output the current permutation and π stop.

We test both the referenced local search and the local search without order in our study. The local search is applied to each generated offspring with a probability P_{ls}, that is, local search is applied if a random number uniformly generated in the range of [0,1] is less than P_{ls}. In addition, the local search is also applied to the best individual after the initialization of the population. Obviously, the previously proposed speed-up procedure is used in the presented local search methods.

4.5. Diversity controlling mechanism

Invariably, as the population of the EDA evolves over generations, its diversity diminishes and the individuals in the population become very similar. This results in search stagnation. To overcome this problem, as did in the literature [45,58], a restart mechanism is applied when the diversity value falls below a given threshold value γ. In the restart mechanism, the 20% best individuals are kept from the current population and the remaining 80% are generated randomly. At the same time, to reduce the computation, the diversity value is calculated at least 100 generations after the algorithm restarts. In addition, we present a very simple method to evaluate the diversity of the population based on both the job order and on similar blocks of jobs in the sequences of the current population as follows:

Step 1. Calculate the matrix $\lfloor\phi_{i,j}\rfloor_{n\times n}$ as

$$[\phi_{i,j}]_{n\times n}=\begin{bmatrix} \phi_{1,1} & \phi_{1,2} & \cdots & \phi_{1,n} \\ \phi_{2,1} & \phi_{2,2} & \cdots & \phi_{2,n} \\ \cdots & \cdots & \ddots & \cdots \\ \phi_{n,1} & \phi_{n,2} & \cdots & \phi_{n,n} \end{bmatrix}$$

where $\phi_{i,j}$ is the number of times that job j appears at position i.

Step 2: Calculate matrix $[\lambda_{j',j}]_{n\times n}$ as follows:

$$[\lambda_{j',j}]_{n\times n}=\begin{bmatrix} - & \lambda_{1,2} & \cdots & \lambda_{1,n}\\ \lambda_{2,1} & - & \cdots & \lambda_{2,n}\\ \cdots & \cdots & \ddots & \cdots\\ \lambda_{n,1} & \lambda_{n,2} & \cdots & - \end{bmatrix}$$

recall that $\lambda_{j',j}$ represents the number of times that job j appears immediately after job j'.

Step 3: Count the number of elements which are larger than zero in $[\phi_{i,j}]_{n\times n}$, and denote it as α.

Step 4: Count the number of elements which are larger than zero in $[\lambda_{j,j'}]_{n\times n}$, and denote it as β.

Step 5: The diversity value of the population div is then computed as follows:

$$div=\left(\frac{\alpha-n}{n\times\min(n,PS-1)}+\frac{\beta-(n-1)}{(n-1)\times\min(n-1,PS-1)}\right)\Big/2$$

Hence, div gives a very simple diversity measure with a value between zero and one. Obviously, the higher the div value is, the more diverse the population is. A value close to one indicates a very diverse population where each job occupies different positions and no similar blocks of jobs exist among the individuals. On the other hand, a value close to zero means that all individuals are very similar or identical. A simple example is given by considering a population of three individuals with four jobs: $\pi(1)=\{1,2,3,4\}$, $\pi(2)=\{2,3,4,1\}$ and $\pi(3)=\{1,4,2,3\}$. Firstly, we calculate $[\phi_{i,j}]_{n\times n}$ and $[\lambda_{j,j'}]_{n\times n}$ as follows:

$$[\phi_{i,j}]_{4\times 4}=\begin{bmatrix}2&1&0&0\\0&1&1&1\\0&1&1&1\\1&0&1&1\end{bmatrix},\quad [\lambda_{j',j}]_{4\times 4}=\begin{bmatrix}-&1&0&1\\0&-&3&0\\0&0&-&2\\1&1&0&-\end{bmatrix}$$

Then we get $\alpha=11$ and $\beta=6$.

Finally, we obtain $div=((11-4)/(4\times\min(4,3-1))+(6-3)/((4-1)\times\min(4-1,3-1)))/2=0.69$.

5. Calibration of the proposed EDA

Considering all previous sections, the proposed EDA method goes as follows:

Step 1: Set the algorithm parameters PS, Q, M, P_{ls}, ε, θ, γ. Let $gen=1$.
Step 2: Initialize the population and evaluate each individual.
Step 3: Perform local search to the best individual in the initial population.
Step 4: Select Q best individuals and estimate the probabilistic model.
Step 5: Sample and generate M offspring from the probabilistic model.
Step 6: Perform local search to each offspring in M with probability P_{ls}.
Step 7: Evaluate the offspring and update the population.
Step 8: Check the diversity of the population if $gen>100$. If the diversity level is less than γ, perform restart procedure, and set $gen=0$; otherwise set $gen=gen+1$.
Step 9: If the stopping criterion is reached, return the best solution found so far and stop; otherwise, go to step 4.

As we can see, the proposed EDA depends on 8 parameters. Therefore, we need to carry out a calibration in order to set them to appropriate values. We first carefully decide the ranges of parameters according to the existing literature, like carried out by Ruiz et al. [45] and by Vallada and Ruiz [58], among many others and also according to our past experience. Then, we conduct a preliminary experiment to determine the levels for each parameter. In the experiment, we try several typical values for each parameter by simply fixing others, and select the best two or three for calibration in our calibration experiment to keep the aforementioned calibrations at a manageable level. Next, we employ a Design of Experiments approach where each parameter is a controlled factor as follows: population size (PS) tested at three levels, 10, 30 and 50. Selection size (Q) tested at two levels, 5 and 10. Offspring number (M) tested at two levels, 5 and 10. Probability to apply local search (P_{ls}) tested at two levels, 0.15 and 0.30. Local search type with two variants, referenced local search and local search without order. Parameter (ε) (generation of new individuals from Section 4.3) tested at two values, 0.7 and 0.9. Parameter (θ) (also from Section 4.3) tested at two values, 2 and 5. Finally, we have the diversity threshold (γ) tested at 0.3 and 0.5 values. This results in a total of $3\times2\times2\times2\times2\times2\times2=384$ different combinations, i.e., 384 different configurations for the proposed EDA. All the 384 configurations are tested in a full factorial experimental design with a termination criterion of maximum elapsed CPU time of $t=50\times n\times(m/2)$ ms. This termination allows for more time as the number of jobs and machines increases, and has been used in Ruiz et al. [45] and by Vallada and Ruiz [58] and by many others. Each algorithm is tested with a small set of 24 randomly generated instances. The number of jobs and machines for each instance are chosen randomly from the following sets $n\in\{10,30,50,70,90,110\}$and $m\in\{5,10,15,20\}$. Following Yoon and Ventura [62] and Tseng and Liao [56], the related data for the instances is given by discrete uniform distributions as follows: $l(j)\in U[1,6]$, $p(k,j)\in U[1,31]$ and $s(k,j',j)\in U[1,31]$. For each instance, 5 difference replicates are run. Therefore, the total number of results is $384\times24\times5=46{,}080$. Two sets of experiments are conducted: one for the idling and another for the no-idling case.

The proposed EDA procedure is coded in Visual C++ 6.0 and all the 384 configurations are run on a cluster of 30 blade servers each one with two Intel XEON E5420 processors running at 2.5 GHz and with 16 GB of RAM memory. Each processor has four cores and the experiments are carried out in virtualized Windows XP machines, each one with one virtualized processor and 2 GB of RAM memory. As a response variable for the experiment, we measure the relative percentage increase (RPI)

$$RPI(c_i)=(c_i-c^*)/c^*\times100 \tag{14}$$

where c_i is the makespan value generated in the ith replication by a given algorithm configuration, and c^* is the best objective value found by any of the algorithm configurations. Note that for this problem there are no known effective exact techniques and comparing against an optimum solution is not possible. Due to the sequence-dependent setup times, lower bounds are extremely weak and the results would be difficult to analyze. Instead of carrying out a comparison against the best solution given by an algorithm, we tried to obtain better solutions by running the best tested method for an extended period of time. This resulted in negligible differences so we preferred to compare algorithms against the best solution given by them.

All results are analyzed by means of the Analysis of Variance (ANOVA) technique, a very powerful statistical approach that allows us to set the different parameters at statistically significant values among the tested ones. This approach has been followed in Ruiz et al. [45], Ruiz and Stutzle [46], Ribas et al. [42], among many others.

The results of both calibration experiments (idling and no-idling) are very similar. All 8 controlled factors (parameters of the proposed DEA) are statistically significant at a 95% confidence level. The ANOVA table with the full results is not shown here due

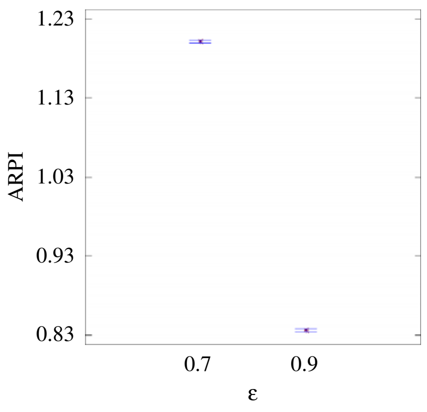

Fig. 5. Means plot and 95% Tukey HSD confidence intervals for the calibration experiment in the idling case, factor ε.

to reasons of space. However, all experimental results are available upon request from the authors. Let us picture just one result for the most significant factor in the idling experiment, which is factor ε, whose means plot and 95% Tukey Honest Significant Difference (HSD) confidence intervals are given in Fig. 5.

As we can see, a level of 0.9 for the factor ε is statistically better (and by a wide margin) than the value 0.7. This means that in the generation of offspring, it is much better to use the proposed probabilistic model than the reference solution.

After the calibration experiments, we set the parameters as follows for both the idling and no-idling cases: $\varepsilon=0.9$, $PS=10$, $\theta=5$, $Q=10$, $\gamma=0.3$, $P_{ls}=0.15$, $M=10$, Local search is referenced local search (factors in order of statistical relevance).

It might be argued that the presented EDA can be further improved by trying consecutive rounds of tuning a few significant parameters and fixing the rest to the best combination found in the above full factorial experiment. We have tried consecutive rounds of tuning by setting ε from 0.85 to 1.0 with a step equal to 0.01 and other parameters unchanged. The experimental results show that the EDA with $\varepsilon=0.95$ produces better results than with $\varepsilon=0.9$. However, these differences are not large (about 0.2%) and have little relevance in reality. Therefore, to avoid the problem of over-calibration, we adopt the parameters calibrated by the previous ANOVA.

6. Computational results and comparisons

Several metaheuristics exist in the literature for solving n-job m-machine lot-streaming flow shop scheduling problems. Although none of them considers sequence-dependent setup times, we have carried out a comprehensive re-implementation and adaptation work of most published related material for comparisons. Marimuthu et al. [27,28,29] presented seven methods including a tabu search (which we denote by TS), simulated annealing with insertion neighborhood (SA_i), simulated annealing with swap neighborhood (SA_s), hybrid genetic algorithm (HGA), ant colony optimization (ACO), threshold accepting with insertion neighborhood (TA_i) and threshold accepting with swap neighborhood (TA_s) to minimize both mak$n \times m$espan and total flow time for an n-job m-machine lot-streaming flow shop problem involving attached setup times. By numerical comparison, the authors claimed that their algorithms were effective and efficient for the problem considered. Tseng and Liao [56] developed a discrete particle swarm optimization (DPSO) algorithm for an n-job m-machine lot-streaming flow shop scheduling problem with the objective to minimize the mean weighted absolute deviation of job completion times from their due dates, and it was demonstrated by the authors that their DPSO algorithm performed much better than the HGA proposed by Yoon and Ventura [63] for solving 900 randomly generated instances. More recently, Pan et al. [33] presented a discrete artificial bee colony (DABC) algorithm for the problem considered by Tseng and Liao [56] and Yoon and Ventura [62], which outperformed the previously commented DPSO and HGA methods. We compare the proposed EDA with the above 9 state-of-the-art algorithms, i.e., TS, SA_i, SA_s, HGA, ACO, TA_i and TA_s by Marimuthu et al. [27,28,29], the DPSO algorithm by Tseng and Liao [56], and the DABC algorithm by Pan et al. [33], for solving the problem considered in this paper. We also compare with a recently presented EDA (denoted as EDA_J) by Jarboui et al. [15], which was a new state-of-the-art algorithm for minimizing the total flow time in the permutation show shop scheduling problem and provided new upper bounds for 49 out of 90 Taillard benchmark instances. Since the above algorithms are not designed for the problem considered here, we adapt them by using the makespan calculation presented in Section 3, including all accelerations, whenever possible. For the proposed EDA in this paper, we also test it without the speed up procedure (denoted as EDA_{nS}) and without local search (denoted as EDA_{nL}), to show the effect of the speed-up and local search procedures.

To test all the methods (13 in total), we employ a completely different benchmark as the one used before for calibration. The reason is simple: testing with the same benchmark used for calibration would lead to biased results. We use 28 different problem sizes $n \times m$, where $n=30$, 50, 70, 90, 110, 130, 150, and $m=5$, 10, 15, 20. For each $n \times m$ combination, 10 different instances are randomly generated. As a result, the benchmark has 280 instances. The related data for the instances is given by the discrete uniform distributions as follows: $l(j) \in U[1,6]$, $p(k,j) \in U[1,31]$ and $s(k,j',j) \in U[1,31]$. All the algorithms were coded in Visual C++ and executed on the same cluster of machines employed for the calibration. For the EDA, we adopt the parameters and operators calibrated in Section 5, whereas for the other algorithms, the parameters are fixed to those given in the literature. Note that calibration is a fine-tuning process and algorithms are not expected to behave entirely different after calibrations.

To make a fair comparison, all the compared algorithms adopt the same maximum elapsed CPU time limit of $t=n \times (m/2) \times \rho$ ms as a termination criterion, where ρ has been tested at three values: 100, 200 and 300. For each of the 280 instances, 5 independent replications are carried out and for each replication, the RPI is calculated. In addition, the average RPI (ARPI) over each problem size and the overall mean ARPI is also calculated as statistics for the solution quality.

Note that there are 13 algorithms, 280 instances and 5 replications for a total of 18,200 results for each value of ρ (54,600 results in total). The comparisons are carried out both for the idling as well as for the no-idling cases.

6.1. Comparison under the no-idling case

The computational results are reported in Tables 1–3. Note that each cell contains the averages of the 5 replicates for each one of the 10 instances of each $n \times m$ combination (50 values averaged at each cell).

It can be easily seen from Table 1 that, for the shortest elapsed CPU time of $\rho=100$, the proposed EDA is the best performer with the lowest ARPI of just 0.25%, which is significantly smaller than all other results. More interestingly, the EDA achieves the best ARPI values for all 28 problem sizes as well. Compared with the EDA, the EDA_{nS} yields much worse ARPI values for all the 28 problem sizes and a much larger overall ARPI value, which suggests that taking advantage of the speed-up method in the proposed EDA is very beneficial. However, EDA_{nS} is still better than all other methods. On the other hand, both EDA and EDA_{nS}

Table 1
Comparison of algorithms at no-idling case ($\rho=100$).

$n\times m$	EDA	EDA$_{nS}$	EDA$_{nL}$	EDA$_J$	DABC	ACO	DPSO	HGA	SA$_i$	SA$_s$	TA$_i$	TA$_s$	TS
30×5	0.11	0.79	2.32	4.34	1.40	1.82	1.65	3.01	3.29	4.72	4.17	5.67	1.50
30×10	0.27	0.85	2.73	4.06	1.33	2.05	1.44	3.52	3.77	5.10	4.88	5.84	1.52
30×15	0.16	0.60	2.71	3.81	1.17	2.17	1.33	3.32	3.97	4.84	5.33	6.35	1.30
30×20	0.16	0.57	2.63	3.44	1.01	2.00	1.12	3.14	3.43	4.81	4.44	5.80	0.95
50×5	0.21	1.81	3.62	6.99	2.11	2.83	4.16	3.85	3.61	4.78	4.37	5.77	1.98
50×10	0.32	1.91	3.65	6.61	2.13	2.75	4.11	4.29	4.07	5.15	5.27	6.24	2.14
50×15	0.33	1.76	3.54	6.62	2.18	2.59	4.26	4.78	4.50	5.70	5.42	6.84	2.12
50×20	0.34	1.81	3.86	6.19	1.91	2.26	3.66	4.21	4.34	5.42	5.54	6.39	2.18
70×5	0.29	2.23	4.20	8.32	2.45	3.24	6.30	3.35	3.22	4.39	3.85	4.99	2.24
70×10	0.31	2.51	4.48	7.80	2.46	3.12	6.15	4.31	4.00	4.92	4.69	6.22	2.57
70×15	0.33	2.35	4.24	7.52	2.32	3.15	5.79	4.53	4.13	4.92	5.56	6.51	2.52
70×20	0.34	2.44	4.24	7.46	2.38	3.13	6.02	4.52	4.30	5.36	5.56	6.44	2.72
90×5	0.23	2.16	3.88	8.58	2.40	3.38	8.12	3.34	2.66	3.98	3.65	4.50	2.61
90×10	0.22	2.21	4.11	8.18	2.33	3.01	7.31	3.62	2.93	4.24	3.79	5.11	2.43
90×15	0.30	2.35	3.97	7.99	2.25	3.02	7.02	4.06	3.61	4.61	4.65	5.47	2.73
90×20	0.32	2.37	4.00	7.65	2.46	3.17	6.94	4.28	3.66	4.81	4.83	5.70	3.06
110×5	0.19	1.92	3.79	8.69	2.28	3.27	9.16	4.40	2.23	3.41	2.97	4.05	3.02
110×10	0.28	2.36	3.79	8.23	2.37	3.41	8.49	3.78	2.99	3.98	3.99	4.77	2.98
110×15	0.29	2.29	3.48	8.16	2.35	3.23	8.10	3.71	2.92	4.32	4.28	5.25	3.33
110×20	0.29	2.30	3.61	7.97	2.30	3.21	7.66	3.86	3.26	4.33	4.35	5.45	3.21
130×5	0.17	1.99	3.56	8.71	2.41	3.31	9.80	5.80	2.22	3.23	2.98	3.96	4.34
130×10	0.24	2.19	3.44	8.93	2.32	3.40	8.96	4.45	2.60	3.87	3.77	4.38	3.89
130×15	0.25	2.19	3.32	7.96	2.34	3.13	8.61	3.96	2.78	3.86	4.03	4.75	3.64
130×20	0.26	2.19	2.97	8.07	2.13	2.93	8.28	3.64	2.75	3.67	3.96	4.84	3.25
150×5	0.15	1.82	3.22	8.91	2.18	3.10	10.31	7.30	1.75	2.93	2.59	3.49	6.38
150×10	0.15	1.99	2.87	8.14	2.14	3.27	9.04	5.12	2.09	3.29	3.23	3.94	4.30
150×15	0.24	2.07	2.83	8.21	2.21	3.32	9.07	4.79	2.60	3.50	3.70	4.48	4.21
150×20	0.21	1.81	2.53	7.91	2.06	3.04	8.47	4.20	2.55	3.55	3.85	4.45	3.61
Average	0.25	1.92	3.48	7.34	2.12	2.94	6.48	4.18	3.22	4.35	4.27	5.27	2.88

Table 2
Comparison of algorithms, no-idling case ($\rho=200$).

$n\times m$	EDA	EDA$_{nS}$	EDA$_{nL}$	EDA$_J$	DABC	ACO	DPSO	HGA	SA$_i$	SA$_s$	TA$_i$	TA$_s$	TS
30×5	0.12	0.62	2.30	3.71	1.20	1.79	1.37	2.92	3.35	4.79	4.24	5.74	1.38
30×10	0.20	0.61	2.56	3.40	1.06	2.07	1.18	3.34	3.79	5.12	4.89	5.86	1.29
30×15	0.11	0.43	2.52	3.43	0.98	2.15	0.92	3.14	3.97	4.84	5.33	6.35	1.12
30×20	0.17	0.44	2.53	3.16	0.88	2.03	0.84	2.98	3.48	4.86	4.49	5.85	0.92
50×5	0.18	1.55	3.50	6.16	1.87	2.43	3.28	3.84	3.71	4.89	4.47	5.88	1.86
50×10	0.31	1.67	3.65	6.02	1.86	2.57	3.34	4.32	4.20	5.28	5.40	6.37	1.93
50×15	0.33	1.43	3.53	5.96	1.89	2.53	3.56	4.67	4.61	5.82	5.53	6.96	1.84
50×20	0.39	1.55	3.98	5.74	1.81	2.38	3.20	4.33	4.57	5.65	5.77	6.62	2.04
70×5	0.23	2.07	4.13	7.71	2.14	3.06	4.78	3.46	3.37	4.54	4.00	5.14	1.95
70×10	0.31	2.25	4.60	7.26	2.24	2.97	5.14	4.45	4.23	5.16	4.93	6.46	2.50
70×15	0.33	2.08	4.32	6.86	2.11	2.84	4.97	4.62	4.37	5.16	5.80	6.75	2.30
70×20	0.37	2.17	4.36	7.05	2.23	2.95	5.36	4.65	4.55	5.61	5.81	6.69	2.56
90×5	0.22	2.03	3.99	8.14	2.11	3.36	6.23	3.27	2.86	4.18	3.85	4.70	2.08
90×10	0.26	2.11	4.32	7.67	2.07	2.89	5.94	3.86	3.22	4.54	4.08	5.41	2.27
90×15	0.34	2.28	4.26	7.71	2.09	2.94	5.99	4.38	3.96	4.96	4.99	5.81	2.56
90×20	0.31	2.26	4.18	7.11	2.28	2.96	5.88	4.50	3.90	5.05	5.07	5.94	2.85
110×5	0.19	1.93	4.04	8.39	2.03	3.34	7.29	3.13	2.48	3.67	3.22	4.31	2.21
110×10	0.32	2.31	4.09	7.94	2.14	3.31	7.20	3.84	3.29	4.29	4.30	5.09	2.61
110×15	0.27	2.19	3.73	7.76	2.13	3.11	7.06	3.92	3.18	4.58	4.55	5.52	3.03
110×20	0.30	2.22	3.89	7.60	2.08	3.01	6.70	4.09	3.55	4.63	4.65	5.75	2.83
130×5	0.16	1.88	3.77	8.36	2.08	3.34	8.23	3.84	2.42	3.44	3.18	4.17	2.74
130×10	0.22	2.13	3.65	8.61	2.05	3.23	7.69	3.41	2.80	4.08	3.98	4.59	2.75
130×15	0.26	2.13	3.58	7.64	2.12	2.94	7.23	3.71	3.04	4.12	4.29	5.02	2.86
130×20	0.28	2.17	3.27	7.65	1.90	2.65	7.04	3.52	3.04	3.96	4.25	5.14	2.88
150×5	0.20	1.89	3.49	8.65	2.06	3.26	9.06	5.00	2.01	3.21	2.86	3.76	3.76
150×10	0.21	2.02	3.16	7.91	2.02	3.20	7.93	3.60	2.38	3.58	3.53	4.24	3.18
150×15	0.22	1.94	3.05	7.82	1.98	3.10	8.10	3.60	2.82	3.73	3.92	4.70	3.17
150×20	0.21	1.90	2.81	7.55	1.91	2.93	7.48	3.47	2.84	3.83	4.13	4.74	2.65
Average	0.25	1.80	3.62	6.89	1.90	2.83	5.46	3.85	3.43	4.56	4.48	5.48	2.36

significantly improve each ARPI value generated by the EDA$_{nL}$, which demonstrates the effectiveness of incorporating a local search into the EDA variant. In other words, the superiority of the proposed EDA should be attributed to the combination of global search and local search with an appropriate balance between exploration and exploitation.

Table 3
Comparison of algorithms, no-idling case ($\rho=300$).

$n \times m$	EDA	EDA_{nS}	EDA_{nL}	EDA_J	DABC	ACO	DPSO	HGA	SA_i	SA_s	TA_i	TA_s	TS
30×5	0.11	0.51	2.24	3.39	1.09	1.78	1.12	2.78	3.36	4.80	4.25	5.75	1.34
30×10	0.19	0.46	2.49	3.12	0.92	2.05	0.91	3.25	3.79	5.12	4.90	5.86	1.22
30×15	0.12	0.38	2.48	3.27	0.84	2.17	0.76	2.99	3.99	4.86	5.36	6.37	1.09
30×20	0.17	0.42	2.52	2.96	0.79	2.04	0.68	2.84	3.50	4.88	4.51	5.87	0.93
50×5	0.21	1.42	3.44	5.86	1.86	2.35	3.07	3.83	3.84	5.02	4.60	6.01	1.90
50×10	0.32	1.48	3.64	5.62	1.79	2.54	3.03	4.29	4.26	5.34	5.47	6.44	1.86
50×15	0.33	1.22	3.50	5.58	1.73	2.48	3.18	4.53	4.66	5.86	5.58	7.00	1.75
50×20	0.37	1.37	3.98	5.38	1.67	2.35	2.83	4.33	4.62	5.70	5.82	6.67	2.03
70×5	0.21	1.94	4.08	7.27	2.01	2.97	4.28	3.48	3.47	4.64	4.09	5.24	1.90
70×10	0.26	2.07	4.59	6.91	2.12	2.84	4.75	4.47	4.30	5.23	5.00	6.54	2.27
70×15	0.33	1.91	4.32	6.39	1.97	2.65	4.70	4.65	4.46	5.25	5.89	6.84	1.98
70×20	0.36	2.00	4.41	6.62	2.13	2.84	5.02	4.71	4.64	5.71	5.91	6.79	2.35
90×5	0.22	1.94	4.09	7.88	1.93	3.33	5.51	3.35	3.00	4.32	3.99	4.84	2.03
90×10	0.24	2.05	4.39	7.39	1.97	2.90	5.33	4.00	3.38	4.70	4.24	5.58	1.96
90×15	0.28	2.12	4.33	7.33	2.02	2.83	5.52	4.43	4.06	5.06	5.10	5.92	2.37
90×20	0.36	2.14	4.34	6.86	2.24	2.91	5.66	4.61	4.09	5.24	5.27	6.14	2.73
110×5	0.19	1.92	4.16	8.16	1.87	3.34	6.42	3.08	2.61	3.80	3.35	4.44	1.88
110×10	0.28	2.17	4.21	7.57	2.02	3.17	6.39	3.95	3.41	4.41	4.42	5.20	2.35
110×15	0.32	2.21	3.94	7.58	2.13	3.12	6.51	4.12	3.40	4.81	4.77	5.75	2.83
110×20	0.27	2.14	4.01	7.36	1.98	2.89	6.11	4.20	3.68	4.76	4.77	5.88	2.65
130×5	0.21	2.02	3.96	8.23	1.95	3.38	7.39	3.23	2.61	3.63	3.38	4.36	2.45
130×10	0.23	2.15	3.75	8.26	1.90	3.12	6.90	3.40	2.91	4.19	4.08	4.70	2.23
130×15	0.23	2.13	3.67	7.39	1.95	2.82	6.66	3.78	3.14	4.21	4.39	5.11	2.31
130×20	0.23	2.22	3.39	7.46	1.82	2.52	6.54	3.64	3.17	4.09	4.38	5.26	2.56
150×5	0.15	1.78	3.57	8.51	1.83	3.27	8.43	3.70	2.08	3.28	2.94	3.84	2.87
150×10	0.16	1.93	3.26	7.69	1.78	3.13	7.13	3.25	2.47	3.68	3.62	4.33	2.73
150×15	0.27	1.99	3.24	7.67	1.89	3.07	7.64	3.64	3.02	3.92	4.12	4.90	2.72
150×20	0.21	1.90	2.91	7.32	1.76	2.87	6.98	3.56	2.95	3.94	4.25	4.85	2.36
Average	0.24	1.71	3.68	6.61	1.78	2.78	4.98	3.79	3.53	4.66	4.59	5.59	2.13

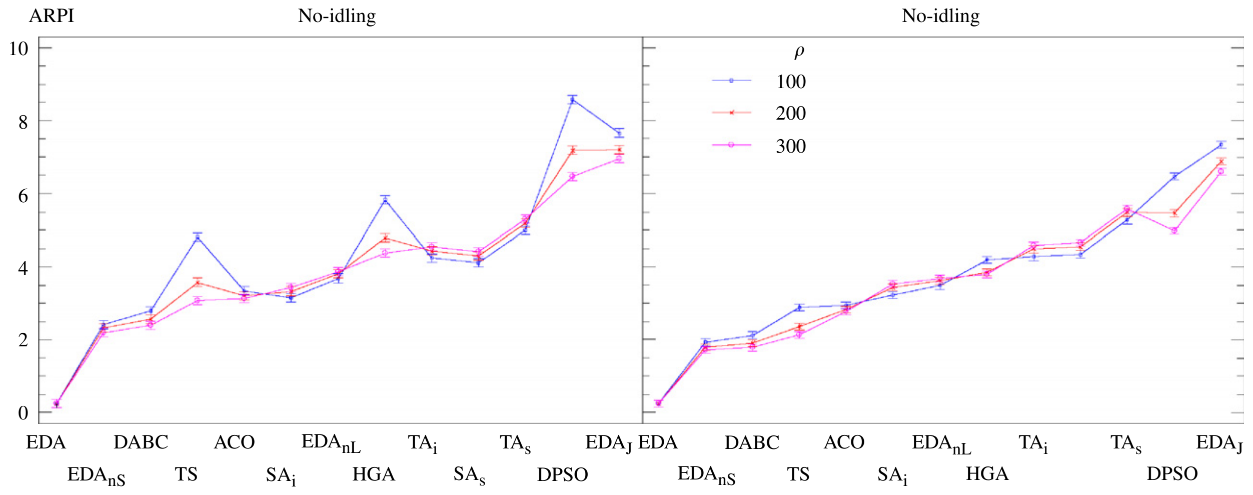

Fig. 6. Means plot and 95% Tukey HSD confidence intervals for the interaction between the algorithms, the maximum elapsed CPU time ρ and the no-idling/idling cases.

The computational results with $\rho=200$ and 300 are reported in Tables 2 and 3, respectively. It is clear from these tables that the results are again favorable.

The presented EDA makes extensive use of some advanced techniques such as an efficient population initialization, a newly designed probabilistic model, a diversity controlling mechanism, and hybridization with local search. These techniques are in favor of the EDA transferring the building blocks of jobs in parents to offspring, maintaining diversity of population, having higher local exploitation ability. In addition, the presented speed-up technology makes the EDA much more effective. Thus the EDA can achieve better performance than the other algorithms at several different levels. Note that in the comprehensive experiments, EDA is compared against other EDA methods (EDA_J) and other GAs. Basically, the differences in efficiency and effectiveness cannot be solely attributed to the fact that we are presenting an EDA method but more precisely to the efficient and effective instantiation of the EDA method for the considered problem.

6.2. Comparison under the idling case

Results for the three different stopping times are given in Tables 4–6. It is clear from the these results that the proposed EDA outperforms the existing methods of the comparison by a considerable margin for the lot-streaming flow shop scheduling problem with setup times to minimize makespan under the idling case. Quite

Table 4
Comparison of algorithms, idling case ($\rho = 100$).

$n \times m$	EDA	EDA$_{nS}$	EDA$_{nL}$	EDA$_J$	DABC	ACO	DPSO	HGA	SA$_i$	SA$_s$	TA$_i$	TA$_s$	TS
30 × 5	0.14	1.37	2.60	5.42	2.07	2.07	2.61	3.63	3.22	4.50	4.41	5.62	1.88
30 × 10	0.24	1.32	2.64	4.95	1.83	2.26	2.35	3.69	3.80	4.86	4.85	5.80	1.85
30 × 15	0.24	1.19	3.04	4.66	1.67	2.14	2.10	3.71	3.94	4.79	5.48	5.75	1.46
30 × 20	0.25	1.03	2.99	4.15	1.40	2.25	1.96	3.40	3.60	4.51	4.97	5.40	1.59
50 × 5	0.21	2.60	3.72	7.98	2.96	3.57	6.96	3.68	3.51	4.76	4.33	5.77	3.16
50 × 10	0.32	2.66	3.90	7.40	2.89	3.44	5.85	4.40	4.26	5.21	5.39	6.20	3.10
50 × 15	0.33	2.57	3.94	6.90	2.81	3.13	5.67	4.49	4.25	5.03	5.54	6.17	2.85
50 × 20	0.34	2.63	4.26	6.71	2.67	2.64	5.24	4.67	4.23	5.01	5.43	5.96	2.73
70 × 5	0.19	2.63	4.13	8.27	3.00	3.42	8.96	4.52	2.85	4.01	3.68	4.67	3.64
70 × 10	0.37	3.14	4.75	8.35	3.28	3.69	8.64	4.64	3.80	4.84	4.58	6.05	3.48
70 × 15	0.32	3.04	4.60	7.80	3.01	3.48	8.14	4.21	3.61	4.78	5.19	5.72	3.39
70 × 20	0.36	2.92	4.39	7.43	2.93	3.41	7.23	4.18	3.84	4.48	5.07	5.62	3.21
90 × 5	0.20	2.69	4.30	8.84	3.23	3.69	11.71	6.52	2.89	4.08	3.43	4.54	5.35
90 × 10	0.20	2.95	3.99	8.62	3.15	3.55	9.94	4.93	2.94	3.99	3.78	4.73	3.62
90 × 15	0.23	2.70	4.19	7.93	3.18	3.46	9.28	4.39	3.50	4.23	4.78	5.04	3.68
90 × 20	0.36	3.01	4.34	7.58	3.17	3.50	8.47	4.22	3.44	4.28	4.74	5.15	3.51
110 × 5	0.16	2.73	4.07	9.01	3.03	3.48	13.13	8.58	2.31	3.40	3.04	3.89	7.56
110 × 10	0.24	2.53	3.64	8.35	2.98	3.89	10.82	6.68	2.88	3.71	4.04	4.76	5.64
110 × 15	0.24	2.65	3.79	8.18	2.94	3.70	10.05	5.55	2.96	3.92	4.14	4.84	4.89
110 × 20	0.27	2.73	3.63	7.86	2.89	3.61	9.22	4.92	2.99	3.92	4.20	4.98	4.37
130 × 5	0.22	2.32	3.81	9.26	3.09	3.47	13.51	10.20	2.42	3.46	2.95	3.80	10.24
130 × 10	0.24	2.61	3.53	8.87	2.96	3.78	11.42	7.96	2.54	3.39	3.76	4.34	7.32
130 × 15	0.23	2.52	3.47	8.24	2.99	3.56	10.39	7.56	2.59	3.51	3.72	4.32	5.97
130 × 20	0.21	2.48	3.30	8.12	2.90	3.55	9.95	6.79	2.50	3.49	4.03	4.54	5.41
150 × 5	0.22	2.06	3.25	9.12	2.90	3.42	13.46	11.72	2.21	3.20	2.90	3.55	12.20
150 × 10	0.21	2.12	2.91	8.59	2.68	3.74	12.01	8.98	2.37	3.18	3.29	4.09	8.77
150 × 15	0.27	2.31	3.07	8.27	2.77	3.94	10.98	8.11	2.52	3.38	3.66	4.42	7.43
150 × 20	0.20	2.28	3.00	7.89	2.74	3.62	10.21	7.07	2.36	3.20	3.57	4.10	6.52
Average	0.25	2.42	3.69	7.67	2.79	3.34	8.58	5.84	3.15	4.11	4.25	4.99	4.82

Table 5
Comparison of algorithms, idling case ($\rho = 200$).

$n \times m$	EDA	EDA$_{nS}$	EDA$_{nL}$	EDA$_J$	DABC	ACO	DPSO	HGA	SA$_i$	SA$_s$	TA$_i$	TA$_s$	TS
30 × 5	0.15	0.99	2.48	4.74	1.67	1.90	2.03	3.48	3.28	4.56	4.47	5.68	1.76
30 × 10	0.19	0.95	2.56	4.27	1.49	2.14	1.90	3.58	3.83	4.89	4.89	5.84	1.43
30 × 15	0.24	0.93	2.94	3.98	1.37	2.15	1.62	3.62	3.99	4.84	5.54	5.80	1.43
30 × 20	0.23	0.82	2.90	3.59	1.19	2.28	1.49	3.27	3.64	4.56	5.01	5.45	1.37
50 × 5	0.20	2.28	3.62	7.34	2.64	3.31	4.93	3.73	3.64	4.90	4.47	5.90	2.64
50 × 10	0.31	2.30	3.77	6.63	2.55	3.08	4.70	4.39	4.38	5.34	5.51	6.33	2.42
50 × 15	0.26	2.09	3.82	6.25	2.34	2.71	4.45	4.38	4.30	5.08	5.59	6.23	2.09
50 × 20	0.43	2.25	4.33	6.12	2.42	2.50	4.41	4.78	4.44	5.23	5.65	6.19	2.33
70 × 5	0.20	2.52	4.25	7.85	2.73	3.33	7.14	3.76	3.11	4.27	3.94	4.94	2.72
70 × 10	0.33	2.89	4.83	7.92	2.96	3.46	6.80	4.69	4.02	5.06	4.79	6.27	3.15
70 × 15	0.30	2.92	4.71	7.36	2.77	3.21	6.38	4.34	3.87	5.05	5.46	5.99	2.90
70 × 20	0.33	2.67	4.39	6.83	2.64	3.10	6.01	4.35	4.05	4.70	5.29	5.84	2.81
90 × 5	0.21	2.64	4.54	8.57	2.95	3.80	9.43	4.77	3.16	4.36	3.70	4.81	3.54
90 × 10	0.18	2.78	4.22	8.15	2.85	3.38	8.32	3.78	3.18	4.23	4.02	4.98	2.65
90 × 15	0.21	2.71	4.39	7.44	2.77	3.30	7.92	3.96	3.72	4.45	5.01	5.26	2.66
90 × 20	0.35	2.92	4.60	7.21	2.80	3.31	7.28	4.11	3.73	4.57	5.03	5.44	3.18
110 × 5	0.18	2.53	4.31	8.65	2.86	3.58	10.17	6.15	2.49	3.63	3.25	4.13	4.77
110 × 10	0.22	2.56	3.91	7.99	2.90	3.89	9.41	4.73	3.14	3.98	4.31	5.03	3.95
110 × 15	0.24	2.62	4.04	7.83	2.74	3.48	8.56	4.07	3.21	4.17	4.39	5.09	3.84
110 × 20	0.24	2.68	3.93	7.54	2.75	3.38	8.17	4.00	3.29	4.22	4.51	5.28	3.46
130 × 5	0.19	2.46	3.99	8.92	2.90	3.61	11.12	7.71	2.42	3.63	3.08	3.98	6.90
130 × 10	0.23	2.78	3.81	8.51	2.90	3.80	9.83	5.99	2.79	3.64	4.02	4.62	5.04
130 × 15	0.30	2.75	3.79	7.90	2.83	3.44	9.55	4.88	2.91	3.83	4.04	4.64	4.25
130 × 20	0.24	2.64	3.55	7.71	2.76	3.30	8.88	4.34	2.75	3.74	4.29	4.80	4.04
150 × 5	0.17	2.16	3.40	8.80	2.72	3.43	11.65	9.32	2.11	3.23	2.86	3.67	9.01
150 × 10	0.17	2.22	3.11	8.14	2.61	3.70	10.20	6.60	2.48	3.39	3.46	4.30	6.02
150 × 15	0.25	2.40	3.28	7.97	2.75	3.72	9.79	6.01	2.70	3.59	3.87	4.63	5.10
150 × 20	0.15	2.40	3.25	7.60	2.75	3.53	9.21	5.24	2.58	3.45	3.83	4.36	4.61
Average	0.24	2.32	3.81	7.21	2.56	3.21	7.19	4.79	3.33	4.31	4.44	5.20	3.57

interestingly, the additional elapsed CPU time does not seem to affect the proposed EDA method. The conclusion is that the presented EDA is capable of reaching good solutions very quickly and stagnates around very good solutions that are probably close to optimal.

6.3. *Statistical assessment of results*

While the results in all previous tables show strong differences between the proposed EDA and all the considered methods, it is

Table 6
Comparison of algorithms, idling case ($\rho = 300$).

$n \times m$	EDA	EDA$_{nS}$	EDA$_{nL}$	EDA$_{J}$	DABC	ACO	DPSO	HGA	SA$_{i}$	SA$_{s}$	TA$_{i}$	TA$_{s}$	TS
30×5	0.15	0.85	2.39	4.43	1.49	1.82	1.73	3.29	3.30	4.59	4.50	5.71	1.61
30×10	0.18	0.77	2.47	3.92	1.29	2.14	1.63	3.51	3.83	4.89	4.89	5.84	1.33
30×15	0.22	0.77	2.87	3.68	1.20	2.16	1.34	3.39	4.00	4.85	5.55	5.81	1.36
30×20	0.21	0.68	2.80	3.26	1.08	2.27	1.29	3.26	3.66	4.57	5.03	5.46	1.27
50×5	0.21	2.06	3.53	6.82	2.38	3.08	4.36	3.72	3.68	4.94	4.51	5.95	2.49
50×10	0.29	2.03	3.64	6.25	2.38	2.94	4.17	4.38	4.42	5.37	5.55	6.36	2.20
50×15	0.33	1.88	3.85	6.04	2.27	2.64	4.02	4.46	4.45	5.23	5.73	6.37	2.19
50×20	0.41	1.92	4.20	5.82	2.16	2.38	3.92	4.71	4.47	5.26	5.68	6.21	1.95
70×5	0.19	2.37	4.27	7.60	2.56	3.33	6.11	3.82	3.22	4.38	4.05	5.05	2.36
70×10	0.32	2.74	4.87	7.58	2.71	3.22	5.97	4.62	4.12	5.16	4.90	6.38	2.90
70×15	0.26	2.62	4.63	7.05	2.54	3.02	5.55	4.38	3.93	5.10	5.51	6.04	2.66
70×20	0.33	2.50	4.42	6.59	2.56	2.98	5.58	4.49	4.20	4.85	5.43	5.99	2.62
90×5	0.30	2.61	4.72	8.38	2.78	3.86	8.22	3.95	3.37	4.57	3.91	5.03	3.04
90×10	0.20	2.63	4.33	7.91	2.67	3.24	7.41	3.81	3.32	4.36	4.15	5.11	2.53
90×15	0.24	2.62	4.56	7.31	2.61	3.24	6.91	4.08	3.93	4.66	5.21	5.47	2.58
90×20	0.34	2.84	4.73	6.99	2.68	3.13	6.59	4.16	3.88	4.72	5.18	5.59	3.08
110×5	0.18	2.50	4.45	8.37	2.79	3.64	9.41	4.90	2.62	3.77	3.39	4.27	4.00
110×10	0.22	2.58	4.05	7.85	2.67	3.83	8.75	4.05	3.29	4.13	4.46	5.18	3.34
110×15	0.26	2.67	4.23	7.66	2.65	3.40	7.74	3.91	3.39	4.36	4.58	5.29	3.43
110×20	0.27	2.67	4.07	7.33	2.53	3.29	7.26	3.97	3.44	4.37	4.66	5.43	3.13
130×5	0.16	2.32	4.05	8.66	2.76	3.59	10.05	6.29	2.46	3.69	3.14	4.04	5.21
130×10	0.22	2.58	3.95	8.22	2.75	3.77	9.06	4.87	2.93	3.78	4.16	4.76	4.23
130×15	0.26	2.51	3.88	7.70	2.66	3.25	8.48	4.06	2.99	3.92	4.12	4.73	3.45
130×20	0.25	2.54	3.71	7.59	2.60	3.12	8.15	3.84	2.91	3.90	4.45	4.95	3.60
150×5	0.15	2.18	3.49	8.65	2.58	3.52	10.56	7.83	2.16	3.33	2.96	3.77	7.14
150×10	0.21	2.25	3.27	8.01	2.58	3.70	9.37	5.44	2.63	3.54	3.62	4.46	4.57
150×15	0.19	2.41	3.37	7.72	2.60	3.53	9.09	4.90	2.78	3.68	3.96	4.72	3.93
150×20	0.18	2.45	3.42	7.40	2.61	3.47	8.45	4.43	2.76	3.63	4.00	4.54	3.87
Average	0.24	2.20	3.87	6.96	2.40	3.13	6.47	4.38	3.43	4.41	4.55	5.30	3.07

still necessary to carry out a statistical experiment to attest if the observed differences are indeed statistically significant. We have carried out a full factorial ANOVA where n, m, instance number, replicate, ρ, the type of algorithm and idling/no-idling factors are considered. There are important statistically significant differences. Fig. 6 shows a three-way interaction between the type of algorithm, the maximum elapsed CPU time factor ρ and idling and no-idling cases. We are now employing a 95% confidence level and we are using Tukey HSD confidence intervals. Note that overlapping intervals denote a statistically insignificant difference in the plotted means. From the figure it is clear that the proposed EDA produces results that are statistically better than all the considered algorithms. It is also shown that the EDA shows statistically insignificant differences with more allotted CPU time. i.e., $\rho = 200$ or $\rho = 300$ result in no additional gains. Most other methods improve results with additional elapsed CPU time.

As a result, we can safely conclude that the proposed EDA is a new effective algorithm for the lot-streaming flow shop scheduling problem with sequence-dependent setup times and makespan criterion in both the idling and no-idling cases.

7. Conclusions

This paper studies the flow shop scheduling problem under the lot-streaming generalization and with sequence-dependent setup times. The studied objective is makespan minimization. This problem has important applications in textile, plastic, chemical, semiconductor, and many other industries where jobs are actually batches of many identical products to be manufactured. A novel estimation of distribution algorithm (EDA) has been proposed for the problem under both the idling and no-idling cases. To the best of our knowledge, this is the first attempt at solving the problem considered, and this is also the first reported application of EDA for solving lot-streaming flow shop scheduling problems. An extensive comparison has been carried out for the proposed EDA against the best existing metaheuristics developed for lot-streaming flow shop problems, as well as against a recently presented EDA for the traditional flow shop problem with total flow time criterion. According to the computational results and statistical analyses, the proposed EDA clearly outperforms all the other considered algorithms by a considerable margin for the lot-streaming flow shop problem with setup times to minimize makespan.

The superiority of the presented EDA is mainly due to the fact that it extensively uses some advanced techniques such as an efficient population initialization, a newly designed probabilistic model, a diversity controlling mechanism, hybridization with local search, and a speed-up procedure. The population initialization mechanism provides an initial population with a high level of quality and diversity. The presented probabilistic model helps in transferring the building blocks of jobs in parents to offspring. The diversity controlling mechanism aims at maintaining the diversity of the population and without it the algorithm stalled after just a few iterations. The hybridization with local search not only enhances the algorithm's local exploitation ability, but also provides an appropriate balance between exploration of the global search and exploitation of the local search. The presented speed-up method improves the search efficiency by a significant margin.

The proposed EDA can be extended to take into account more realistic aspects of the lot-streaming problem, such as the existence of due dates, machine eligibility, parallel machines, multiple objectives and many others. Late work criteria are being actively studied nowadays, as the study of Sterna [52] attests. The proposed EDA can also be generalized to solve other combinatorial optimization problems including the hybrid flow shop, job shop, the traveling salesman or complex scheduling problems as those studied in Manaa and Chu [25], Ruiz-Torres et al. [47] or Gribkovskaia et al. [13]. Specific hybrid flow shops as the ones approached in Samarghandi and ElMekkawy [49] or in Besbes et al. [5] are equally interesting. Some other single machine problems with many added constraints, as the one studied in Valente and Schaller [57] seem a promising venue of research for the application of the techniques studied in this paper. Of course, each problem would need special tailoring and experimentation and this is the basis for future research.

Acknowledgments

This research is partially supported by the National Science Foundation of China (60874075, 70871065), and Science Foundation of Shandong Province in China under Grant BS2010DX005, and Postdoctoral Science Foundation of China under Grant 20100480897. Rubén Ruiz is partially funded by the Spanish Ministry of Science and Innovation, under the project "SMPA—Advanced Parallel Multiobjective Sequencing: Practical and Theoretical Advances" with reference DPI2008-03511/DPI and by the IMPIVA—Institute for the Small and Medium Valencian Enterprise, for the project OSC with references IMIDIC/2008/137, IMIDIC/2009/198 and IMIDIC/2010/175.

References

[1] Allahverdi A, Gupta JND, Aldowaisan T. A review of scheduling research involving setup considerations. Omega—International Journal of Management Science 1999;27(2):219–39.
[2] Allahverdi A, Ng CT, Cheng TCE, Kovalyov MY. A survey of scheduling problems with setup times or costs. European Journal of Operational Research 2008;187(3):985–1032.
[3] Allahverdi A, Soroush HM. The significance of reducing setup times/setup costs. European Journal of Operational Research 2008;187(3):978–84.
[4] Baker KR, Jia D. A comparative-study of lot streaming procedures. Omega—International Journal of Management Science 1993;21(5):561–6.
[5] Besbes W, Teghem J, Loukil T. Scheduling hybrid flow shop problem with non-fixed availability constraints. European Journal of Industrial Engineering 2010;4(4):413–33.
[6] Bukchin J, Tzur M, Jaffe M. Lot splitting to minimize average flow-time in two-machine flow-shop. IIE Transactions 2002;34(11):953–70.
[7] Çetinkaya FC. Lot streaming in a two-stage flow shop with set-up, processing and removal times separated. Journal of the Operational Research Society 1994;45(12):1445–55.
[8] Chang JH, Chiu HN. A comprehensive review of lot streaming. International Journal of Production Research 2005;43(8):1515–36.
[9] Edis RS, Ornek MA. A tabu search-based heuristic for single-product lot streaming problems in flow shops. International Journal of Advanced Manufacturing Technology 2009;43(11-12):1202–13.
[10] Feldmann M, Biskup D. Lot streaming in a multiple product permutation flow shop with intermingling. International Journal of Production Research 2008;46(1):197–216.
[11] Framinan JM, Leisten R, Rajendran C. Different initial sequences for the heuristic of Nawaz, Enscore and Ham to minimize makespan, idletime or flowtime in the static permutation flowshop sequencing problem. International Journal of Production Research 2003;41(1):121–48.
[12] Garey MR, Johnson DS, Sethi R. The complexity of flowshop and jobshop scheduling. Mathematics of Operations Research 1976;1(2):117–29.
[13] Gribkovskaia IV, Kovalev S, Werner F. Batching for work and rework processes on dedicated facilities to minimize the makespan. Omega—International Journal of Management Science 2011;38(6):522–7.
[14] Huang RH, Yang CL. Solving a multi-objective overlapping flow-shop scheduling. International Journal of Advanced Manufacturing Technology 2009;42(9-10):955–62.
[15] Jarboui B, Eddaly M, Siarry P. An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problems. Computers & Operations Research 2009;36(9):2638–46.
[16] Kalir AA, Sarin SC. A near-optimal heuristic for the sequencing problem in multiple-batch flow-shops with small equal sublots. Omega—International Journal of Management Science 2001;29(6):577–84.
[17] Kalir AA, Sarin SC. Constructing near optimal schedules for the flow-shop lot streaming problem with sublot-attached setups. Journal of Combinatorial Optimization 2003;7(1):23–44.
[18] Kim K, Jeong IJ. Flow shop scheduling with no-wait flexible lot streaming using an adaptive genetic algorithm. International Journal of Advanced Manufacturing Technology 2009;44(11-12):1181–90.
[19] Kropp DH, Smunt TL. Optimal and heuristic models for lot splitting in a flow-shop. Decision Sciences 1990;21(4):691–709.
[20] Kumar S, Bagchi TP, Sriskandarajah C. Lot streaming and scheduling heuristics for m-machine no-wait flowshops. Computers & Industrial Engineering 2000;38(1):149–72.
[21] Larrañaga P, Lozano JA. Estimation of distribution algorithms. A new tool for evolutionary computation. Boston: Kluwer Academic Publishers; 2002.
[22] Liu JY. Single-job lot streaming in m-1 two-stage hybrid flowshops. European Journal of Operational Research 2008;187(3):1171–83.
[23] Liu SC. A heuristic method for discrete lot streaming with variable sublots in a flow shop. International Journal of Advanced Manufacturing Technology 2003;22(9–10):662–8.
[24] Liu S-C, Chen E-C, Liu HT. A heuristic method for multi-product variable lot streaming in a flow shop. Journal of the Chinese Institute of Industrial Engineers 2006;23(1):65–79.
[25] Manaa A, Chu C. Scheduling multiprocessor tasks to minimise the makespan on two dedicated processors. European Journal of Industrial Engineering 2010;4(3):265–79.
[26] Marimuthu S, Ponnambalam SG. Heuristic search algorithms for lot streaming in a two-machine flowshop. International Journal of Advanced Manufacturing Technology 2005;27(1-2):174–80.
[27] Marimuthu S, Ponnambalam SG, Jawahar N. Tabu search and simulated annealing algorithms for scheduling in flow shops with lot streaming. Proceedings of the Institution of Mechanical Engineers Part B—Journal of Engineering Manufacture 2007;221(2):317–31.
[28] Marimuthu S, Ponnambalam SG, Jawahar N. Evolutionary algorithms for scheduling m-machine flow shop with lot streaming. Robotics and Computer-Integrated Manufacturing 2008;24(1):125–39.
[29] Marimuthu S, Ponnambalam SG, Jawahar N. Threshold accepting and ant-colony optimization algorithms for scheduling m-machine flow shops with lot streaming. Journal of Materials Processing Technology 2009;209(2):1026–41.
[30] Martin CH. A hybrid genetic algorithm/mathematical programming approach to the multi-family flowshop scheduling problem with lot streaming. Omega-International Journal of Management Science 2009;37(1):126–37.
[31] Mühlenbein H, Paass G. From recombination of genes to the estimation of distributions I binary parameters. In: Voignt H-M, editor. Proceedings of the fourth international conference on parallel problem solving from nature, vol. 1141. Berlin: Springer; 1996. p. 178–87.
[32] Nawaz M, Enscore Jr. EE, Ham I. A heuristic algorithm for the m machine, n job flowshop sequencing problem. Omega-International Journal of Management Science 1983;11(1):91–5.
[33] Pan Q-K, Suganhan PN, Tasgetiren MF, Chua TJ. A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. Information Sciences 2011;181(12):2455–68.
[34] Pan Q-K, Tasgetiren MF, Liang YC. A discrete differential evolution algorithm for the permutation flowshop scheduling problem. Computers & Industrial Engineering 2008;55(4):795–816.
[35] Pan Q-K, Wang L. A novel differential evolution algorithm for no-idle permutation flow-shop scheduling problems. European Journal of Industrial Engineering 2008;2(3):279–97.
[36] Pan Q-K, Wang L, Qian B. A novel differential evolution algorithm for bi-criteria no-wait flow shop scheduling problems. Computers & Operations Research 2009;36(8):2498–511.
[37] Potts CN, Baker KR. Flow-shop scheduling with lot streaming. Operations Research Letters 1989;8(6):297–303.
[38] Rad SF, Ruiz R, Boroojerdian N. New high performing heuristics for minimizing makespan in permutation flowshops. Omega—International Journal of Management Science 2009;37(2):331–45.
[39] Rajendran C, Ziegler H. Two ant-colony algorithms for minimizing total flowtime in permutation flowshops. Computers & Industrial Engineering 2005;48(4):789–97.
[40] Reiter S. System for managing job-shop production. Journal of Business 1966;39(3):371–93.
[41] Ríos-Mercado RZ, Bard JF. Heuristics for the flow line problem with setup costs. European Journal of Operational Research 1998;110(1):76–98.
[42] Ribas I, Companys R, Tort-Martorell X. An iterated greedy algorithm for the flowshop scheduling with blocking. Omega—The international Journal of Management Science 2011;39(3):293–301.
[43] Ruiz R, Allahverdi A. Some effective heuristics for no-wait flowshops with setup times to minimize total completion time. Annals of Operations Research 2007;156(1):143–71.
[44] Ruiz R, Maroto C. A comprehensive review and evaluation of permutation flowshop heuristics. European Journal of Operational Research 2005;165(2):479–94.
[45] Ruiz R, Maroto C, Alcaraz J. Two new robust genetic algorithms for the flowshop scheduling problem. Omega—International Journal of Management Science 2006;34(5):461–76.
[46] Ruiz R, Stutzle T. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. European Journal of Operational Research 2007;177(3):2033–49.
[47] Ruiz-Torres AJ, Ho JH, Ablanedo-Rosas JH. Makespan and workstation utilization minimization in a flowshop with operations flexibility. Omega—International Journal of Management Science 2011;39(3):273–82.
[48] Salhi A, Vázquez Rodríguez JA, Zhang Q. An estimation of distribution algorithm with guided mutation for a complex flow shop scheduling problem. In: Thierens D, editor. Proceedings of the ninth annual conference on Genetic and evolutionary computation, vol. 1. New York: The Association for Computing Machinery; 2010. p. 570–6.

[49] Samarghandi H, ElMekkawy TY. An efficient hybrid algorithm for the two-machine no-wait flow shop problem with separable setup times and single server. European Journal of Industrial Engineering 2011;5(2):111–31.
[50] Sarin SC, Jaiprakash P. Flow Shop Lot Streaming. New York: Springer; 2007.
[51] Sriskandarajah C, Wagneur E. Lot streaming and scheduling multiple products in two-machine no-wait flowshops. IIE Transactions 1999;31(8):695–707.
[52] Sterna M. A survey of scheduling problems with late work criteria. Omega-International Journal of Management Science 2011;39(2):120–9.
[53] Taillard E. Some efficient heuristic methods for the flow-shop sequencing problem. European Journal of Operational Research 1990;47(1):65–74.
[54] Trietsch D, Baker KR. Basic techniques for lot streaming. Operations Research 1993;41(6):1065–76.
[55] Truscott WG. Production scheduling with capacity-constrained transportation activities. Journal of Operations Management 1986;6(3-4):333–48.
[56] Tseng CT, Liao CJ. A discrete particle swarm optimization for lot-streaming flowshop scheduling problem. European Journal of Operational Research 2008;191(2):360–73.
[57] Valente JMS, Schaller JE. Improved heuristics for the single machine scheduling problem with linear early and quadratic tardy penalties. European Journal of Industrial Engineering 2010;4(1):99–129.
[58] Vallada E, Ruiz R. Genetic algorithms with path relinking for the minimum tardiness permutation flowshop problem. Omega—International Journal of Management Science 2010;38(1-2):57–67.
[59] Vickson RG. Optimal lot streaming for multiple products in a two-machine flow shop. European Journal of Operational Research 1995;85(3):556–75.
[60] Vickson RG, Alfredsson BE. Two-machine and three-machine flow shop scheduling problems with equal sized transfer batches. International Journal of Production Research 1992;30(7):1551–74.
[61] Wang L, Pan QK, Suganthan PN, Wang WH, Wang YM. A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems. Computers & Operations Research 2010;37(3):509–20.
[62] Yoon SH, Ventura JA. An application of genetic algorithms to lot-streaming flow shop scheduling. IIE Transactions 2002;34(9):779–87.
[63] Yoon SH, Ventura JA. Minimizing the mean weighted absolute deviation from due dates in lot-streaming flow shop scheduling. Computers & Operations Research 2002;29(10):1301–15.
[64] Zhang W, Yin CY, Liu JY, Linn RJ. Multi-job lot streaming to minimize the mean completion time in m-1 hybrid flowshops. International Journal of Production Economics 2005;96(2):189–200.