An iterated greedy heuristic for no-wait flow shops with sequence dependent setup times, learning and forgetting effects

Xiaoping Li[a,b,*], Zhi Yang[a,b], Rubén Ruiz[c], Tian Chen[a,b], Shaochun Sui[d]

[a]*School of Computer Science and Engineering, Southeast University, Nanjing 211189, China*
[b]*Key Laboratory of Computer Network and Information Integration, Ministry of Education, Nanjing, 211189, China*
[c]*Grupo de Sistemas de Optimización Aplicada, Instituto Tecnológico de Informática, Ciudad Politécnica de la Innovación, Edifico 8G, Acc. B. Universitat Politècnica de València, Camino de Vera s/n, 46021, València, Spain*
[d]*Production Management, AVIC Chengdu Aircraft Industrial (Group) CO., LTD., Chengdu 610091, China*

Abstract

In this paper we address the problem of the sequence dependent setup times no-wait flowshop with learning and forgetting effects to minimize total flowtime. Due to the NP-Hard nature of this problem, several simple metaheuristic methods are presented in this paper. A position-based learning and forgetting effects model is constructed where the processing times of operations vary according to the positions of the jobs in the schedule. An accelerated neighbourhood construction procedure is presented. Given the the simplicity and excellent performance shown in flowshop scheduling problems, an iterated greedy heuristic is studied. To improve the quality of the solutions, the proposed method employs local search heuristics based on Variable Neighbourhood Descent. The presented procedure is compared with some existing algorithms for similar problems on an exhaustive computational campaign. Comprehensive experimental results show that the proposal obtains the best performance among the compared methods by a wide and statistically significant margin.

Keywords: Scheduling, Sequence dependent setup times, Learning and forgetting effects, No-wait Flowshop

1. Introduction

There are a lot activities undertaken by humans in many industries, especially in manufacturing environments. Generally, learning occurs when similar tasks are done repeatedly and

*Corresponding author: Xiaoping Li, Professor of the School of Computer Science and Engineering, Southeast University, Nanjing 211189, China. Tel.& Fax: 86-25-52090916.

Email addresses: xpli@seu.edu.cn (Xiaoping Li), yang_1990_zhi@163.com (Zhi Yang), rruiz@eio.upv.es (Rubén Ruiz), ctnjpub@163.com (Tian Chen), suishaochun@vip.163.com (Shaochun Sui)

learning effects decrease processing times. At the same time, forgetting occurs when workers relearn the process after an interruption for a batch of tasks or machine maintenance, etc. In contrast to learning effects, forgetting effects increase processing times. Job deterioration is caused by the forgetting effect [WC07b]. In many real-life situations, the phenomena of the learning effect and deteriorating jobs occurs simultaneously [WC07a].

The sequence dependent setup times no-wait flowshop is one of the constrained flowshop scheduling problems which has been applied widely in different industries, such as metal, plastic, textile, chemical and semiconductor [NZ14, GS88, HS96]. This problem is also motivated by concepts such as just-in-time and zero inventory in modern manufacturing systems. In no-wait flowshop problems, the different operations of each job have to be processed without interruption between consecutive machines, i.e., the start of a job must be delayed on the first machine, if necessary, so that the job need not wait for processing on subsequent machines. Setup time is the time required to prepare a device, machine, process, or system for it to be ready to process a job. Most of the time, setup times are separated from processing times because they are non-negligible in real industrial environments.

In this paper, we consider the no-wait flowshop problem with sequence dependent setup times as well as with learning and forgetting effects to minimize the total flowtime. To the best of our knowledge, this problem has not been investigated yet. This problem involves several challenges: (i) Besides learning effects caused by human activities, forgetting effects result from the sequence dependent setup times. Their parameters (the learning index, the forgetting index, etc.) have a great influence on the learning and forgetting effects. It is hard to design an appropriate learning and forgetting effect model which is suitable for practical no-wait flowshop applications with sequence dependent setup times. (ii) Because of learning and forgetting effects, the processing times of jobs' operations do not remain unchanged any more as they do in traditional no-wait flowshops [LWW08] and they change with jobs's positions in a schedule. (iii) Because of changing processing times, the existing fast objective increment computing methods (such as those in [LWW08]) are not suitable for the problem under study, which drives us to derive new objective computing properties and construct new accelerate operators. (iv) According to the obtained properties and operators, it is necessary to develop effective methods for the considered problem to meet practical requirements. The main contributions of this paper are summarized below:

- We construct a position-based learning and forgetting effects model for no-wait flowshop problems with sequence dependent setup times and tune their appropriate parameters.
- Objective increment properties are deduced for the problem under study. Three accelerate neighbourhood construction heuristics are presented.
- An iterated greedy heuristic framework is proposed. We design and calibrate the components and their parameters over a large number of instances.

The remainder of the paper is organized as follows. The state of the art of the problem under study is reviewed in Section 2. Section 3 describes the considered problem, adjacent job distances and the learning and forgetting model. In order to calculate solutions faster, properties for adjacent job distances and fast neighbourhood construction methods are illustrated in Section 4. Section 5 details the proposed iterated greedy heuristic. Experimental results are provided and analyzed in Section 6, followed by the conclusion in Section 7.

2. Related Work

The no-wait flowshop is one of the most studied variants of the regular flowshop problem with literally hundreds of papers published in the literature. Recently, [NM16] published a review about constructive heuristics. The problem version with the additional consideration of sequence dependent setup times has received a lot of recent attention [AJRA14, AS13, NA14]. There are usually two types of setup times: sequence independent setup times, and sequence dependent setup times. Makespan, total flowtime and total tardiness are three commonly studied objectives.

Wright [Wri36] observed that the processing time of a job is shortened if it is scheduled later in a sequence. This phenomenon is called the learning effect [Bis99]. There are three types of learning effect models commonly used in the literature: position-based, experience-based and the sum-of-processing-time based. The position-based learning model is the most common, which can be further classified into the job-independent model and the job-dependent one. Biskup [Bis99] formulated the processing time of a job as $p_{i,r} = p_i r^{\alpha}$, which is a function of the scheduled position r. p_i is the normal processing time of job i and α is the learning index, which depends on the learning rate LR. Mosheiov [Mos01] extended Biskup's learning model to $p_{i,r} = p_i r^{\alpha_i}$ where there is a different learning index $\alpha_i \leq 0$ for job i. A position dependent linear learning function $p_{i,r} = p_i - b_i \min\{r-1, g_i\}$ was given by Cheng and Wang [CW00], where $b_i > 0$ is the linear learning rate of job i and g_i is the learning threshold.

Many exact methods, heuristics, and meta-heuristics have been proposed for flowshop problems with learning effects. Chung and Tong [CT12] adapted two well-known heuristics for a bi-criteria scheduling problem in an m-machine permutation flowshop with varied learning effects on different machines. Wang et al. [WZZ+13] considered the flowshop scheduling with a truncated position-based learning effect to minimize one of the six regular performance criteria (total completion time, makespan, total weighted completion time, discounted total weighted completion time, sum of the quadratic job completion times, and maximum lateness). Heuristics were presented along with the worst-case bound being analyzed. Lee and Chung [LC13] proposed a branch-and-bound method and two heuristics for a permutation flowshop scheduling problem with learning effect to minimize the total tardiness. Vahedi Nouri, et al. [VNFR13] proposed a hybrid firefly-simulated annealing algorithm for the flowshop problem with learning effect and flexible maintenance activities to minimize the sum of tardiness and maintenance costs. Wang and Wang [WW14] adapted two well-known heuristics for the flowshop scheduling with a general exponential learning effect to minimize makespan, total (weighted) completion time, total weighted discounted completion time and sum of the quadratic job completion times respectively. In addition, some metaheuristics have been proposed for sequence-dependent setup time flowshop problems. Behnamian and Zandieh [BZ13] developed a hybrid metaheuristic, which hybrids Particle Swarm Optimization, Simulated Annealing and Variable Neighborhood Search, for the sequence-dependent setup time hybrid flowshop with the position-based learning effect to minimize earliness and tardiness. Pargar and Zandieh [PZ12] considered the sequence-dependent setup time hybrid flowshop scheduling with learning effect to minimize the weighted sum of makespan and

total tardiness, for which a meta-heuristic approach WFA (water flow-like algorithm) was investigated.

There are some works focusing on both learning and forgetting effects. Lee [Lee04] first considered learning and forgetting effects simultaneously for single-machine scheduling problems to minimize makespan, total flowtime and total lateness, and introduced two learning and forgetting effects models $p_{i,r} = \alpha_i t r^a$ and $p_{i,r} = (p_0 + \alpha_i t) r^a$. Wang and Cheng [WC07a] proposed two general learning and forgetting effects models $p_{j,r}(t) = \alpha_j (b + ct) r^a$ and $p_{i,j,r}(t) = \alpha_{i,j}(b + ct) r^a$ for single machine problems and flowshop problems to minimize four performance measurements: makespan, total completion time, total weighted completion time, and maximum lateness. Wang [Wan06] constructed the model with the learning and forgetting effects $p_{j,r} = (\alpha_j + \beta t) r^a$, which was applied to the machine scheduling problems for minimizing makespan, the (weighted) sum of completion times and maximum lateness. Several single machine and flowshop problems were shown to be polynomially solvable. Wu et al. [YC08] considered a two-machine total completion time flowshop problem. Actual job processing time functions depend on both the processed jobs and a control parameter. A branch-and-bound and a genetic heuristic-based algorithm were proposed. Wang et al. [WJCW12] considered a two-machine flowshop scheduling problem to minimize the makespan with deterioration and learning effects. Dominance properties and two lower bounds were derived to speed up the elimination process of the proposed branch-and-bound algorithm. Two heuristic algorithms were developed to obtain near-optimal solutions. Yin et al. [YLHZ12] introduced a general scheduling model for single-machine scheduling problems, which considered the effects of position-dependent learning and time-dependent deterioration simultaneously. Some approximation algorithms were presented and the worst case error bound was analyzed for the considered single-machine scheduling problems. However, to the best of our knowledge, both learning and forgetting effects have neither been considered in general flowshops nor in no-wait flowshops as yet.

Many heuristics have been proposed for flowshop scheduling problems. However, the Iterated Greedy (IG) proposed by Ruiz and Stützle [RS07] is the most popular because it is simple to implement and is highly effective for flowshops. Recently, Pan and Ruiz [PR12] showed that IG obtains very good results for the permutation flowshop scheduling problem with total flowtime minimization, which is similar to the considered problem. Other authors have proposed highly effective IG-based methods for similar flowshop problems. For example, Ribas et al. [RCTM11] present an IG for a flowshop problem with blocking constraints. Only recently this algorithm was improved by Pan et al. [PWS+13] but also by using elements of the IG methodology. Ruiz and Stützle show in [RS08] an IG method for a flowshop with sequence dependent setup times. Pan et al. [PWZ08] proposed an IG for a no-wait flowshop. All these papers show state-of-the-art IG methods for related variants of the problem we are studying in this paper, so it seems a promising venue of research to consider IG methods for the problem at hand.

3. SDST-NWFSP with learning and forgetting effects

The sequence dependent setup times no-wait flowshop problem (SDST-NWFSP) with learning and forgetting effects consists of a set of n jobs $\mathbb{J} = \{J_1, J_2, \ldots, J_n\}$ to be processed on a set of m machines $\mathbb{M} = \{M_1, M_2, \ldots, M_m\}$. Each job J_i is processed successively on the m machines in the same order and without interruption. Once a job starts processing on the first machine, its successive operations cannot be interrupted before completion, either on or between machines. Each job can only be processed on one machine at the same time and each machine processes only one operation at any time. The normal processing time of each operation $O_{i,j}$ of J_i on machine M_j is $p_{i,j}$, which is increased or decreased by the learning or forgetting effect. A period of setup time is needed before $O_{i,j}$, which depends on the job scheduling sequence.

Let $\pi(n) = (\pi_{[0]}, \pi_{[1]}, \ldots, \pi_{[n]})$ be a schedule of the n jobs. $\pi_{[k]} \in \mathbb{J}$ is the k-th $(k = 1, \ldots, n)$ job in $\pi(n)$ and $\pi_{[0]}$ is a dummy job with zero processing time and zero setup time. All the permutations of the n jobs are denoted as Ω, i.e., $\Omega = \{\pi(n)\}$. Let $B_{i,k,r}$, $E_{i,k,r}$ and $p_{i,k,r}$ be the start time, finish time and the processing time of $O_{i,j}$ when J_i is located at the r^{th} position of schedule $\pi(n)$ respectively. $s_{i,j,k}$ is the setup time between adjacent operations $O_{i,k}$ and $O_{j,k}$. The target is to find the optimum schedule π^* with $TFT(\pi^*) = \min_{\pi \in \Omega}\{TFT(\pi)\}$.

Similar to the no-wait flowshop with makespan minimization in [LWW08], $TFT(\pi)$ can be calculated by the weighted sum of job-pair distances of all the adjacent jobs because of the no-wait characteristic. Let $D^{\pi}_{i,j,r}$ be the distance between the completion times of adjacent jobs J_i and J_j on the last machine M_m when J_i and J_j are located at the r^{th} and $(r+1)^{th}$ positions of π with learning and forgetting effects respectively. $D_{i,j,r}$ is exemplified in Figure 1.

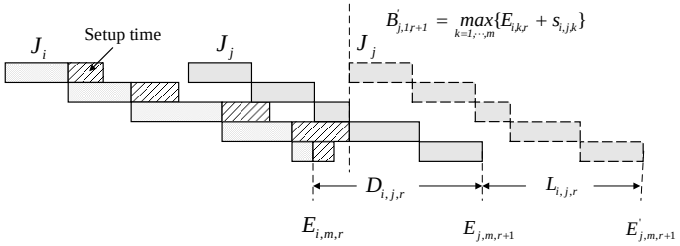

Figure 1: Distance between two adjacent jobs.

To calculate $D_{i,j,r}$, assume that the setup operation of J_j at position $r+1$ is conducted immediately after J_i at position r $(0 \le r < n)$ and that J_j could start on the first machine only after all of its setup operations have finished, i.e. $B'_{j,1,r+1} = \max_{k=1,\ldots,m}\{E_{i,k,r} + s_{i,j,k}\}$. It is necessary to shift J_j leftwards until the start time of J_j on any machine (not necessarily on the last machine) is equal to the finish time of its setup operation on that machine. Because of the no-wait characteristic, $B_{i,k,r} = B_{i,1,r} + \sum_{j=1}^{k} p_{i,j,r} - p_{i,k,r}$ and $E_{i,k,r} = B_{i,k,r} + p_{i,k,r} = B_{i,1,r} + \sum_{j=1}^{k} p_{i,j,r}$, we obtain:

$$B'_{j,k,r+1} = B'_{j,1,r+1} + \sum_{h=1}^{k} p_{j,h,r+1} - p_{j,k,r+1} = \max_{h=1,\ldots,m}\left\{E_{i,h,r} + s_{i,j,h}\right\} + \sum_{h=1}^{k} p_{j,h,r+1} - p_{j,k,r+1}$$

$$= \max_{h=1,\ldots,m}\Big\{B_{i,1,r} + \sum_{u=1}^{h} p_{i,u,r} + s_{i,j,h}\Big\} \quad + \sum_{h=1}^{k} p_{j,h,r+1} - p_{j,k,r+1}$$

$$E^{'}_{j,k,r+1} = B^{'}_{j,k,r+1} + p_{j,k,r+1} = \max_{h=1,\ldots,m}\Big\{B_{i,1,r} + \sum_{u=1}^{h} p_{i,u,r} + s_{i,j,h}\Big\} + \sum_{h=1}^{k} p_{j,h,r+1}$$

Then the maximum shifting distance $L_{i,j,r}$ is

$$L_{i,j,r} = \min_{k=1,\ldots,m}\Big\{B^{'}_{j,k,r+1} - (E_{i,k,r} + s_{i,j,k})\Big\}$$
$$= \min_{k=1,\ldots,m}\Big\{\max_{h=1,\ldots,m}\Big\{\sum_{u=1}^{h} p_{i,u,r} + s_{i,j,h}\Big\}\Big\} + \sum_{h=1}^{k} p_{j,h,r+1} - p_{j,k,r+1} - (\sum_{h=1}^{k} p_{i,h,r} + s_{i,j,k})\Big\}$$

$D_{i,j,r}$ can be computed as follows:

$$D_{i,j,r} = E_{j,m,r+1} - E_{i,m,r} = E^{'}_{j,m,r+1} - L_{i,j,r} - E_{i,m,r} = \max_{h=1,\ldots,m}\Big\{\sum_{u=1}^{h} p_{i,u,r} + s_{i,j,h}\Big\} + \sum_{h=1}^{m} p_{j,h,r+1} -$$
$$\min_{k=1,\ldots,m}\Big\{\max_{h=1,\ldots,m}\Big\{\sum_{u=1}^{h} p_{i,u,r} + s_{i,j,h}\Big\} + \sum_{h=1}^{k} p_{j,h,r+1} - p_{j,k,r+1} - (\sum_{h=1}^{k} p_{i,h,r} + s_{i,j,k})\Big\} - \sum_{h=1}^{m} p_{i,h,r}$$
$$= \max_{k=1,\ldots,m}\Big\{\sum_{h=k}^{m}(p_{j,h,r+1} - p_{i,h,r}) + p_{i,k,r} + s_{i,j,k}\Big\} \tag{1}$$

Equation (1) implies that $D_{i,j,r}$ depends on the processing time of J_i and J_j as well as on the setup time $s_{i,j,k}$. In other words, $D_{i,j,r}$ is unrelated to either the processing times or setup times of the other jobs in the sequence. $D_{i,j,r}$ can be calculated in $O(m)$ steps. However, because of the learning and forgetting effects, $p_{j,h,r+1}$ and $p_{i,h,r}$ are closely related to the positions of jobs J_j and J_i. This is completely different from the traditional no-wait flowshops without learning or forgetting effects, e.g., in [LWW08]. The total flowtime of π now can be computed as follows:

$$TFT(\pi) = \sum_{i=0}^{n-1}(n-i)D^{\pi}_{[i],[i+1],i} \tag{2}$$

In this paper, we construct a position-based learning and forgetting effects model. If job J_i is scheduled at the r^{th} position in a sequence, then the actual processing time of operation $O_{i,j}$ is determined by

$$p_{i,j,r} = p_{i,j} - p_{i,j}\{1-(r+1)^{-\alpha}\}^{\mu} + \gamma\{1-(\beta r+1)e^{-\beta r}\} \times p_{i,j}\{1-(r+1)^{-\alpha}\}^{\mu}$$

where α $(0 \le \alpha \le 1)$ is the learning index, β $(0 \le \beta \le 1)$ the forgetting index, γ the control coefficient and μ the inflection point respectively.

Different learning and forgetting parameter values are suitable for different applications [LWS04, WJCW12, YC08, TCW11]. In order to obtain appropriate parameter values to the application concerned in this paper, we experiment with the values of the four parameters in the learning and forgetting effects model with $\alpha = \{0.45, 0.55, 0.65, 0.75, 0.85\}$,

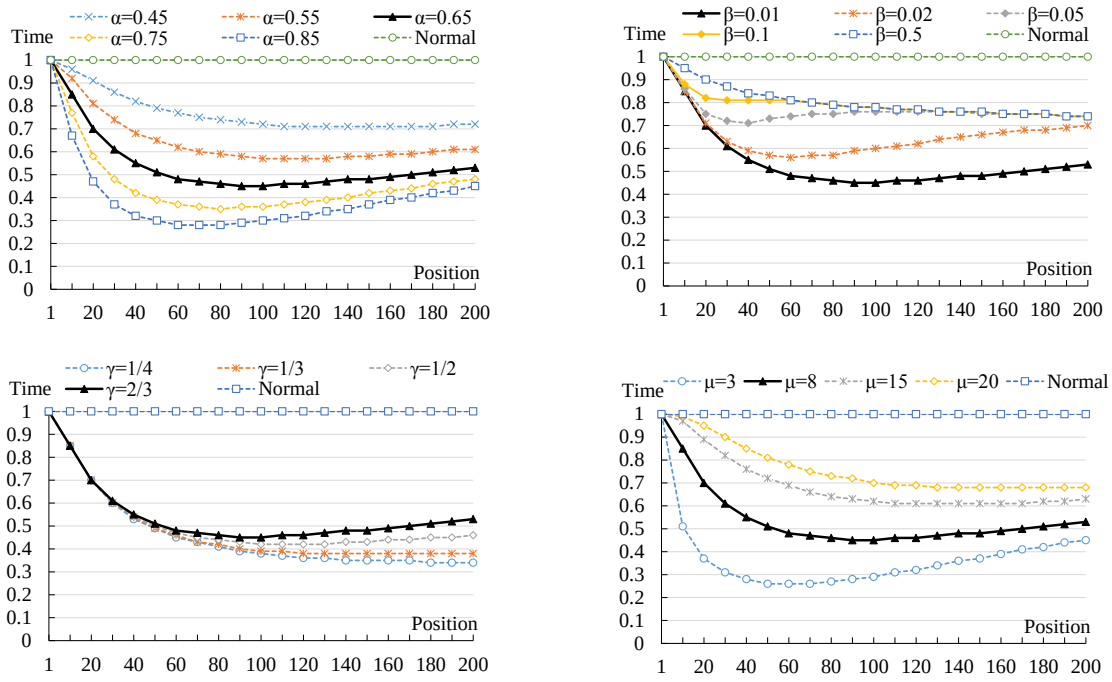

Figure 2: The four parameters of the learning and forgetting effects model.

$\beta = \{0.01, 0.02, 0.05, 0.1, 0.5\}$, $\gamma = \{\frac{1}{4}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}\}$, $\mu = \{3, 8, 15, 20\}$ respectively. To clearly show the learning and forgetting effects, we demonstrate the ratio of the actual processing time to the normal processing time for each parameter tested. In the experiments, 200 jobs are processed on 20 machines. To evaluate the effects of each parameter on the processing times, the other three parameters are fixed. For example, $\beta = 0.01$, $\gamma = \frac{2}{3}$ and $d = 8$ are combined when we test α. The behavior of the four parameters of the learning and forgetting effects model are depicted in Figure 2.

From Figure 2, it can be observed that the larger the α, the smaller the processing time, i.e., α exerts a great influence on the learning effect of the processing time. Likewise, the greater the β, the smaller the processing time. A bigger μ demonstrates a later position of the inflection point. The processing time decreases quickly when the position is small. However, it increases slowly when the position becomes bigger. Therefore, the best combination of the four parameters are $\alpha = 0.65$, $\beta = 0.01$, $\gamma = 2/3$ and $\mu = 8$ for the constructed position-based learning and forgetting effect model in terms of Figure 2.

4. Fast Neighbourhood Construction Methods

4.1. Objective Increment Property

Among all the components of a search algorithm for a combinatorial optimization problem, the objective evaluation operator is usually the most time-consuming. The efficiency of the search process could be improved considerably by reducing the evaluation time of objective

values. In this paper, we introduce the objective increment property for the considered problem. For simplicity, we denote $\xi^{\pi}_{(i,j)} = D^{\pi}_{[i-1],[j],i-1} + D^{\pi}_{[j],[i+1],i} - D^{\pi}_{[i-1],[i],i-1} - D^{\pi}_{[i],[i+1],i}$ and $\eta^{\pi}_{(i,j)} = D^{\pi}_{[j-1],[i],j-1} + D^{\pi}_{[i],[j+1],j} - D^{\pi}_{[j-1],[j],j-1} - D^{\pi}_{[j],[j+1],j}$.

Theorem 1 *By swapping $\pi_{[i]}$ and $\pi_{[j]}$ $(0 < i < j \leq n)$, the total flowtime increment*

$$\delta^{(\pi)}_{(i,j)} = \begin{cases} (n-i)\xi^{\pi}_{(i,j)} + (n-j)\eta^{\pi}_{(i,j)} + D^{\pi}_{[i-1],[j],i-1} - D^{\pi}_{[i-1],[i],i-1} + D^{\pi}_{[j-1],[i],j-1} - D^{\pi}_{[j-1],[j],j-1} & j > i+1 \\ (n-i)\xi^{\pi}_{(i,j)} + (n-j)\eta^{\pi}_{(i,j)} + (n-i)(D^{\pi}_{[i+1],[i],i} + D^{\pi}_{[i],[i+1],i}) & j = i+1 \end{cases}$$

Proof (i) If $0 < i < i+1 < j \leq n$, there is at least one job between $\pi_{[i]}$ and $\pi_{[j]}$. The sequence obtained is $\pi' = (\pi_{[0]}, \pi_{[1]}, \ldots, \pi_{[i-1]}, \pi_{[j]}, \pi_{[i+1]}, \ldots, \pi_{[j-1]}, \pi_{[i]}, \pi_{[j+1]}, \ldots, \pi_{[n]})$ by swapping $\pi_{[i]}$ and $\pi_{[j]}$. Because of the no-wait characteristic and the learning and forgetting effects, $D_{i,j,r}$ depends not only on the processing times and setup times of J_i and J_j but also on the position r of J_i in the schedule, i.e., the same pairs of jobs located at the same position have identical distances. Therefore, $D^{\pi'}_{[k],[k+1],k} = D^{\pi}_{[k],[k+1],k}$ for all $k = 0, \ldots, n-1$ except that $D^{\pi'}_{[i-1],[i],i-1} = D^{\pi}_{[i-1],[j],i-1}$, $D^{\pi'}_{[i],[i+1],i} = D^{\pi}_{[j],[i+1],i}$, $D^{\pi'}_{[j-1],[j],j-1} = D^{\pi}_{[j-1],[i],j-1}$ and $D^{\pi'}_{[j],[j+1],j} = D^{\pi}_{[i],[j+1],j}$.

According to Equation (2), we obtain the total flowtime increment:

$$\begin{aligned} \delta^{\pi}_{(i,j)} &= TFT(\pi') - TFT(\pi) \\ &= \Big\{(n-i+1)D^{\pi}_{[i-1],[j],i-1} + (n-i)D^{\pi}_{[j],[i+1],i} + (n-j+1)D^{\pi}_{[j-1],[i],j-1} + (n-j)D^{\pi}_{[i],[j+1],j}\Big\} - \\ &\quad \Big\{(n-i+1)D^{\pi}_{[i-1],[i],i-1} + (n-i)D^{\pi}_{[i],[i+1],i} + (n-j+1)D^{\pi}_{[j-1],[j],j-1} + (n-j)D^{\pi}_{[j],[j+1],j}\Big\} \\ &= (n-i)\xi^{\pi}_{(i,j)} + (n-j)\eta^{\pi}_{(i,j)} + D^{\pi}_{[i-1],[j],i-1} - D^{\pi}_{[i-1],[i],i-1} + D^{\pi}_{[j-1],[i],j-1} - D^{\pi}_{[j-1],[j],j-1} \end{aligned}$$

(ii) If $0 < i < j \leq n$ and $j = i+1$, the obtained sequence is $\pi' = (\pi_{[0]}, \pi_{[1]}, \ldots, \pi_{[i-1]}, \pi_{[j]}, \pi_{[i]}, \pi_{[j+1]}, \ldots, \pi_{[n]})$ by swapping $\pi_{[i]}$ and $\pi_{[j]}$. $D^{\pi'}_{[k],[k+1],k} = D^{\pi}_{[k],[k+1],k}$ for all $k = 0, \ldots, n-1$ except that $D^{\pi'}_{[i-1],[i],i-1} = D^{\pi}_{[i-1],[j],i-1}$, $D^{\pi'}_{[i],[i+1],i} = D^{\pi}_{[j],[i],i}$ and $D^{\pi'}_{[j],[j+1],j} = D^{\pi}_{[i],[j+1],j}$. According to Equation (2), we obtain the total flowtime increment

$$\begin{aligned} \delta^{\pi}_{(i,j)} &= TFT(\pi') - TFT(\pi) \\ &= \Big\{(n-i+1)D^{\pi}_{[i-1],[j],i-1} + (n-i)D^{\pi}_{[j],[i],i} + (n-j)D^{\pi}_{[i],[j+1],j}\Big\} - \Big\{(n-i+1)D^{\pi}_{[i-1],[i],i-1} + \\ &\quad (n-i)D^{\pi}_{[i],[j],i} + (n-j)D^{\pi}_{[j],[j+1],j}\Big\} \\ &= (n-i)\xi^{\pi}_{(i,j)} + (n-j)\eta^{\pi}_{(i,j)} + (n-i)(D^{\pi}_{[i+1],[i],i} + D^{\pi}_{[i],[i+1],i}) \end{aligned}$$ □

The time complexity of computing total flowtime with the objective increment is only $O(m)$ while without the objective increment it is $O(mn)$ according to Equation (2).

4.2. Neighbourhood Construction

NEH [NEH83] insertion is an effective local search operator which is usually used to construct the neighbourhood of a sequence. However, adjacent job distances are sequence dependent for the considered problem. The objective function value of a newly constructed

sequence (a neighbour) needs to be recalculated. The time complexity of the insertion operator can be decreased using the above objective increment property. In this paper we propose accelerated heuristics for constructing the neighbourhood of a sequence. The computation time of local search operators with the objective increment property is therefore widely reduced.

4.2.1. Accelerated Forward Swap

After a new sequence $\pi^{(0)}$ is constructed by inserting job J_k into the first position of an n-job sequence π, the neighbourhood of $\pi^{(0)}$ is constructed by sequentially swapping J_k with the job to the right. Among the $n+1$ sequences, the best is returned. In the swaps, the above objective increment property is applied to accelerate the objective calculation for each sequence. The process is called Accelerated Forward Swap (AFS), and the function value is illustrated as follows.

- $\pi^{(0)}$ is constructed by inserting job J_k between $\pi_{[0]}$ and $\pi_{[1]}$ of π, i.e., $\pi^{(0)}_{[1]} = J_k$, $\pi^{(0)}_{[2]} = \pi_{[1]}, \ldots,$ $\pi^{(0)}_{[n+1]} = \pi_{[n]}$. Distances $D^{\pi^{(0)}}_{[i],[i+1],i}$ $(0 \le i \le n)$ are computed by Equation (1) with time complexity $O(nm)$ for all the jobs. Therefore $TFT(\pi^{(0)})$ is obtained by Equation (2) with time complexity $O(nm)$.

- Based on $\pi^{(0)}$, $\pi^{(1)}$ is constructed by swapping $\pi^{(0)}_{[1]}$ (or J_k) with $\pi^{(0)}_{[2]}$. According to Theorem 1, $TFT(\pi^{(1)})$ can be obtained directly by $TFT(\pi^{(0)}) + \delta^{\pi^{(0)}}_{(1,2)}$ with time complexity $O(m)$.

- Based on $\pi^{(i-1)}$ $(2 \le i \le n)$, sequence $\pi^{(i)}$ is constructed by swapping $\pi^{(i-1)}_{[i]}$ (or J_k) with $\pi^{(i-1)}_{[i+1]}$. The time complexity of computing $TFT(\pi^{(i)}) = TFT(\pi^{(i-1)}) + \delta^{\pi^{(i-1)}}_{(i,i+1)}$ is $O(m)$.

Among the $n+1$ constructed sequences, the best is returned. AFS is formally described in Algorithm 1.

Algorithm 1: Accelerated Forward Swap (AFS)

Input: Sequence π and job J_k
Output: The best sequence π^* among the $n+1$ neighbours

```
1 begin
2     Generate π^(0) by inserting J_k between π_[0] and π_[1];
3     for i = 0 to n do
4         Calculate D^{π^(0)}_{[i],[i+1],i} by Equation (1);
5     Compute TFT(π^(0)) by Equation (2);
6     for i = 1 to n do
7         Construct π^(i) from π^(i-1) by swapping π^(i-1)_[i] and π^(i-1)_[i+1];
8         Compute TFT(π^(i)) by TFT(π^(i-1)) + δ^{π^(i-1)}_{(i,i+1)};
9     return π* = arg min_{i=0,...,n} {TFT(π^(i))}.
```

The time complexity of AFS is mainly determined by Steps 3 and 6, both of which have a time complexity of $O(nm)$. Therefore the time complexity of AFS is also $O(nm)$.

4.2.2. Accelerated Backward Swap

Accelerated Backward Swap (ABS) is the opposite of AFS. A new sequence $\pi^{(0)}$ is constructed by appending job J_k to an n-job sequence π. The neighbourhood of $\pi^{(0)}$ is constructed by sequentially swapping J_k with the job to the left. Among the $n+1$ sequences, the best is returned. The process of ABS is briefly demonstrated as follows.

- $\pi^{(0)}$ is constructed by appending J_k to the end of π. Since there is no job-position change between π and $\pi^{(0)}$ for the first n jobs, $D^{\pi^{(0)}}_{[i],[i+1],i} = D^{\pi}_{[i],[i+1],i}$ for all $i = 0, \ldots, n-1$. We just calculate $D^{\pi^{(0)}}_{[n],k,n}$ using Equation (1) with time complexity $O(m)$.
- $\pi^{(j)}$ $(j = 1, \ldots, n)$ is constructed from $\pi^{(j-1)}$ by just swapping $\pi^{(j-1)}_{[n+2-j]}$ (or J_k) with its left job $\pi^{(j-1)}_{[n-j+1]}$.

Among the constructed $n+1$ sequences, the best one π^* is returned. ABS is formally described in Algorithm 2.

Algorithm 2: Accelerated Backward Swap (ABS)

Input: Sequence π and job J_k
Output: The best sequence π^* among the $n+1$ neighbours

1 **begin**
2 Generate $\pi^{(0)}$ by appending J_k to π;
3 Calculate $D^{\pi^{(0)}}_{[n],k,n}$ using Equation (1);
4 Compute $TFT(\pi^{(0)})$ by Equation (2);
5 **for** $j = 1$ **to** n **do**
6 Construct $\pi^{(j)}$ from $\pi^{(j-1)}$ by swapping $\pi^{(j-1)}_{[n+2-j]}$ and $\pi^{(j-1)}_{[n-j+1]}$;
7 Compute $TFT(\pi^{(j)})$ by $TFT(\pi^{(j-1)}) + \delta^{\pi^{(j-1)}}_{(n-j+1,n-j+2)}$;
8 **return** $\pi^* = \arg\min_{i=0,\ldots,n} TFT(\pi^{(i)})$.

The time complexity of ABS is mainly determined by Step 5, which is $O(nm)$. Therefore the time complexity of ABS is again $O(nm)$ which is equal to that of AFS. However, the computation time of ABS is roughly much less than that of AFS because ABS only needs one distance computation (with time complexity $O(m)$) while AFS calculates $n+1$ distances (with time complexity $O(nm)$).

4.2.3. Insertion-based Neighbourhood Construction

The traditional insertion method (used in NEH [NEH83] and in RZ [RZ97]) can also be applied to construct the neighbourhood of a sequence. This is called Insertion-based Neighbourhood Construction (INC). A job is inserted into all possible slots and the best sequence is selected. When job J_k is inserted into the j^{th} $(j = 1, \ldots, n)$ slot of π, all the processing times of the jobs $\pi_{[\ell]}$ $(\ell = j, \ldots, n)$ will change because of the learning and forgetting effects. Therefore, $\sum_{j=0}^{n} \sum_{\ell=j}^{n} m$ computations are needed to obtain the objective function values of the newly constructed sequences, i.e., the time complexity of INC is $O(mn^2)$.

Therefore, ABS is the fastest heuristic among the three proposed methods, which will be adopted in the following algorithms. Note that all formulas to calculate the total flowtime in the sequence dependent setup times no-wait flowshop with learning and forgetting effects, as well as all shown computational improvements have been incorporated in the developed algorithm, which, as a result, considers learning and forgetting effects.

5. The Proposed Algorithm

As mentioned in Section 2, we adopt the IG framework, which consists of three basic phases: Initial Sequence Construction, Local Search, Destruction and Reconstruction (D&R). Based on the IG framework, Iterated Greedy heuristics are developed for the considered problem. The framework of the proposed IG is depicted in Algorithm 3. The different operators are explained in the following sections.

Algorithm 3: Iterated Greedy (IG)

begin
  $\pi_0 \leftarrow$ Initial Sequence Construction (ISC);
  $\pi_0 \leftarrow \text{LocalSearch}(\pi_0)$;
  $\pi^* \leftarrow \pi_0$;
  $Temp = T \times \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} p_{i,j}}{n \times m \times 10}$ /* Parameter T will be calibrated later. */
  while *(Termination condition not satisfied)* **do**
    $\pi \leftarrow \text{MDR}(\pi_0)$; /* Modified Destruction & Reconstruction */
    $\pi^c \leftarrow \text{LocalSearch}(\pi)$;
    $\pi \leftarrow \text{AcceptanceCriterion}(\nu, \pi_0, \pi, \pi^c, \pi^*)$;
    $\pi_0 \leftarrow \pi$;
  return π.

5.1. Initial Sequence Construction

Because of the similarity between the studied problems, we modify the initial solution construction method developed by Pan and Ruiz [PR13] and integrate it with the heuristic introduced by Rajendran [Raj93] to generate initial sequences of the considered problem. Seed π^s is generated by sorting all jobs in non-descending order of $\sum_{j=1}^{m}(m-j+1)\times p_{i,j}$, where $p_{i,j}$ is the normal processing time of operation $O_{i,j}$. A new initial solution is constructed by an iterative procedure. An index l starts from 1. A sequence π^l is generated by a two-step process: (i) an n-job schedule $\pi(n)$ is constructed recursively from $\pi(1) = (\pi^s_{[0]}, \pi^s_{[l]})$. $\pi(k)$ is constructed by inserting jobs $\pi^s_{[i]}$ $(i = 1, \ldots, n; i \neq l)$ to $\pi(k-1)$ using the ABS. (ii) π^l is produced by applying the local search method proposed in Section 4.3 to $\pi(n)$. The Initial Sequence Construction (ISC) is formally described in Algorithm 4.

Obviously, the differences between the ISC and the NEH insertion lie in two aspects: (i) ISC adopts the non-descending order of $\sum_{j=1}^{m}(m-j+1)\times p_{i,j}$ to generate the seed while

NEH uses the non-ascending order of the total normal processing times of the jobs; (ii) ISC conducts the ABS while NEH performs the traditional one-job insertion. In terms of Theorem 1, the time complexity of ISC is $O(mn^3)$ since at most n sequences are generated which is equal to the $O(mn^3)$ that would be needed to apply the NEH.

Algorithm 4: Initial Sequence Construction (ISC)

```
begin
    Seed π^s is generated by sorting all jobs with the non-descending order of
      Σ_{j=1}^{m} (m − j + 1) × p_{i,j};
    l ← 1;
    repeat
        k ← 1;
        π(k) ← (π^s_{[0]}, π^s_{[l]});
        for i = 1 to n do
            if i ≠ l then
                k ← k + 1;
                Construct π(k) by inserting job π^s_{[i]} to π(k − 1) using ABS;
        π^l ← LS(π(n));
        l ← l + 1;
    until (CPUTime > mn × 10^{−3} seconds) or (l > n);
    return π* = arg min_{1≤i≤l} {TFT(π^i)}.
```

5.2. Local Search

In this paper, a variable neighborhood descent (VND) is presented for the Local Search to improve intensification of the proposed IG. The commonly used neighborhood structures are $R_d(\pi)$: π' is generated by randomly removing a block of $d+1$ (in this paper $1 \leq d \leq 9$) consecutive jobs from sequence π and reinserting it into the best slot of π. $R_1(\pi)$ was adopted in [LC09, GPSL13]. The variable neighborhood descent (VND) is formally described in Algorithm 5. The VND with parameter d is denoted as LS_d.

5.3. Destruction & Reconstruction

The Local Search process enhances intensification of the search algorithm. However, the balance between intensification and diversification is crucial to avoid being trapped into local optimum [BR03]. The Destruction & Reconstruction (DR) process is carried out on the sequence to improve the diversification of the search process. Sequence π is destructed by randomly selecting and removing k different jobs. Two subsequences π^R and π^D denote the removed k jobs and the remaining $n-k$ jobs respectively. The jobs of π^R are ordered as they were extracted from π. In this paper, a modified Destruction & Reconstruction (MDR) is proposed. Both MDR and the DR developed by Ruiz and Stützle [RS07, RS08] have the same destruction but different reconstruction. When a new job sequence π' is reconstructed,

Algorithm 5: Variable Neighborhood Search (VND)

Input : Parameter d

begin
  Set $l \leftarrow 1$;
  while $l \leq d$ **do**
    Find the best solution $\pi^{'}$ by exploring neighbourhood $R_l(\pi)$;
    if $TFT(\pi^{'}) < TFT(\pi)$ **then**
      $\pi \leftarrow \pi^{'}$, $l \leftarrow 1$;
    else
      $l \leftarrow l + 1$;
  return π.

all jobs of π^R are sequentially reinserted back into π^D during the reconstruction process of DR. Only one job is tried using NEH in every iteration of the DR reconstruction. However, all jobs in π^R are tried using ABS in every iteration of the MDR reconstruction. The MDR is depicted in Algorithm 6. The time complexity of MDR is $O(mnk^2)$.

Algorithm 6: Modified Destruction & Reconstruction (MDR)

begin
  Construct π^R and π^D by randomly removing k jobs from π;
  for $j = 1$ **to** k **do**
    $TFT(\pi^t) \leftarrow \infty$;
    for $i = 1$ **to** $k - j + 1$ **do**
      Construct $\pi^{(i)}$ by inserting job $\pi^R_{[i]}$ to π^D using ABS;
      if $TFT(\pi^t) > TFT(\pi^{(i)})$ **then**
        $\pi^t \leftarrow \pi^{(i)}$, $TFT(\pi^t) \leftarrow TFT(\pi^{(i)})$;
        $\ell \leftarrow i$;
    Remove $\pi^R_{[\ell]}$ from π^R, $\pi^D \leftarrow \pi^t$;
  $\pi \leftarrow \pi^D$;
  return π.

5.4. Acceptance Criterion

Let π_0 be the obtained solution by Local Search on the initial solution, π denotes the reconstructed solution using MDR and π^c represents the solution found by the local search on π. There are three cases for the acceptance criterion: (i) $\nu = 0$. π is replaced by π_0, i.e., π_0 is compared against π^c. (ii) $\nu = 1$. π is compared against π^c. (iii)$\nu = 2$. π is replaced by π_0 if it is worse than π_0, i.e., the best between π_0 and π is compared against π^c. The acceptance

criterion used by Ruiz and Stützle [RS07] is also adopted in this paper. The acceptance criterion process is formally described in Algorithm 7.

Algorithm 7: AcceptanceCriterion($\nu, \pi_0, \pi, \pi^c, \pi^*$)

```
begin
    if ν = 0 then
        π ← π_0;
    if ν = 2 then
        π ← best solution between π_0 and π;
    if TFT(π^c) < TFT(π) then
        π ← π^c;
    else
        Generate a random number λ ∈ [0, 1];
        if λ < e^{-(TFT(π^c)-TFT(π))/Temp} then
            π ← π^c;
    if TFT(π) < TFT(π^*) then
        π^* ← π;
    return π^*.
```

In terms of the acceptance criterion, there are three variants of MDR: MDR_0, MDR_1 and MDR_2. They are constructed by calling Acceptance Criterion with $v = 0$, $v = 1$ and $v = 2$, respectively, after the Local Search.

The final proposed IG is given in Algorithm 8.

Algorithm 8: Proposed Iterated Greedy (IG)

```
begin
    π_0 ← Initial Sequence Construction (ISC, Algorithm 4);
    π_0 ← LocalSearchVND(π_0), (Algorithm 5);
    π^* ← π_0;
    Temp = T × (Σ_{i=1}^{n} Σ_{j=1}^{m} p_{i,j}) / (n × m × 10) while (Termination condition not satisfied) do
        π ← MDR(π_0); Modified Destruction & Reconstruction, (Algorithm 6);
        π^c ← LocalSearchVND(π), (Algorithm 5);
        π ← AcceptanceCriterion(ν, π_0, π, π^c, π^*), (Algorithm 7);
        π_0 ← π;
    return π.
```

6. Experimental evaluations

We calibrate the parameters and components of the proposed IG, which are experimentally determined, before comparing the proposed IG with existing IG methods and heuristics for similar scheduling problems. All algorithms are coded in Java and run on an Intel(R) Core(TM) i7-4770 CPU @3.40GHz computer with 8GB RAM on Windows Server 2008 R2 standard. The termination criterion is set to a maximum computation time of $(n \times m \div 2) \times t$ milliseconds as is now usual in the flowshop scheduling literature where t is a parameter.

6.1. Parameter and Component Calibration

The termination condition parameter is $t = 20$, i.e., the computation time is limited to $(n \times m \div 2) \times 20$ milliseconds for all combinations in the calibration experiment. Once the tests have been conducted, we calculate the effectiveness of an algorithm on an instance by the relative percentage deviation (RPD). Let $V_i(H)$ be the solution of instance i obtained by algorithm H and V_i^* be the best solution for i. RPD is defined as

$$RPD = \frac{V_i(H) - V_i^*}{V_i^*} \times 100\% \tag{3}$$

In a similar way to the experiments given in [RS07], we use a total of 17×4=68 groups of randomly generated calibration instances with non-controllable factors n and m, where n has 17 levels {20, 50, 80, 110, ..., 470, 500} and m has 4 levels {5,10,15,20}. For every combination of n and m, we have five replicates. Therefore, there are $5 \times 68 = 340$ calibration instances in total. The processing times and setup times of jobs are uniformly distributed in the interval [1,99]. There are two parameters k and T in the proposed IG. Analogously to the tested integer values in [RS07], k is tested at 7 levels $\{2, 3, 4, 5, 6, 7, 8\}$. There are three values for $T \in \{0, 0.25, 0.5\}$. For the four components of the proposed IG, there are three variants for constructing the initial solution, four for the D&R strategy, and 11 for the local search (9 new constructed local searches LS_d $(d = 1, 2, \ldots, 9)$, IG_RS_{LS} of [RS07] and the no local search case). Since the D&R strategy MDR_0 is identical to MDR_2 for the no local search case, we only test one of the two combinations, i.e., we only test 43, not $4 \times 11 = 44$ combinations of the two components. Therefore, there are $7 \times 3 \times 3 \times 43 = 2709$ combinations in total. However, it is too time-consuming to test all $2709 \times 340 = 921060$ treatments. We observe that the 9 newly constructed local search methods ($LS_1 \sim LS_9$) interact mainly with MDRs in the IGs and we calibrate them separately.

We analyze the results by the Analysis of Variance technique (ANOVA) which is a very robust parametric technique. A number of hypotheses should be ideally met by the experimental data. Among these, the main three are (in order of importance): independence of the residuals, homoscedasticity or homogeneity of the factor's levels variance and normality in the residuals of the model. Apart from a slight non-normality in the residuals, we can accept all hypotheses easily. The response variable in the experiments is the RPD for each algorithm in every instance.

6.1.1. Local Search Determination

According to the experiments, we perform 5 replicates for each combination of non-controllable factors n and m (340 instances in total), and fix $k = 4$, $T = 0.5$, ISC to construct initial solutions and MDR_1 for the destruction and reconstruction. The means plot and 95% confidence level Tukey HSD (honest significant difference) intervals for d is depicted in Figure 3. HSD is a single-step multiple comparison statistical test, usually used in conjunction with ANOVA to find which averages are actually statistically different from one another.

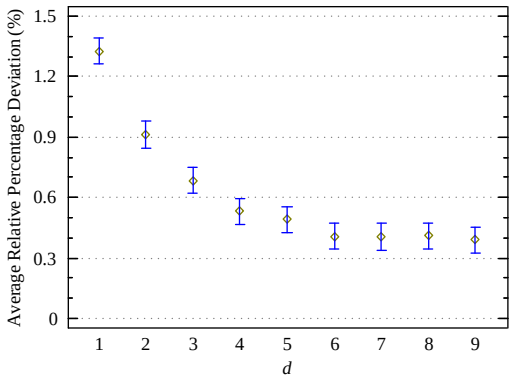

Figure 3: Means plot and 95% confidence level Tukey HSD intervals for d.

Figure 3 illustrates that the observed differences are statistically significant when $d < 6$ and no statistically significant differences are observed when $d \geq 6$ with 95% confidence level Tukey HSD intervals. Therefore, LS_6 is adopted in the following experiments.

6.1.2. Parameters and components

Based on the determined LS_6, there are three local search variants at present: LS_6, IG_RS_{LS} and $NoLS$ (no local search). Since the D&R strategy MDR_0 is identical to MDR_2 for the $NoLS$ case, we just test one of the two combinations, i.e., we only test 11, not $4 \times 3 = 12$ combinations of the two components. Therefore, there are only $7 \times 3 \times 3 \times 11 = 693$ treatments and $693 \times 340 = 235620$ results for calibrating parameters and components. The means plots and 95% confidence level Tukey HSD intervals for k and T are depicted in Figures 4 and 5 respectively.

From Figure 4, it can be observed that the differences are statistically significant when $k \leq 5$ and there is no statistically significant difference when k=6,7,8. It follows that the proposed IG has the minimal ARPD when $k = 5$. Therefore, we use $k = 5$ in the following experiments. Figure 5 implies that the differences are statistically significant for $T = 0$ and $T \neq 0$. There is a tendency towards the difference becoming smaller with higher values of T, e.g., the difference is less than 0.1% between $T = 0.25$ and $T = 0.5$. In other words, the difference is not statistically significant and the algorithm obtains less RPD when $T = 0.5$. Therefore, we set T to 0.5 for the IG proposed in this paper.

Since there is no existing algorithm for the problem under study, we compare the proposed IG heuristic against some algorithms for similar flowshop scheduling problems. In this paper, IG_RS_{LS} [RS07] and IGX [XZL12] are adapted to the problem being studied. There are also

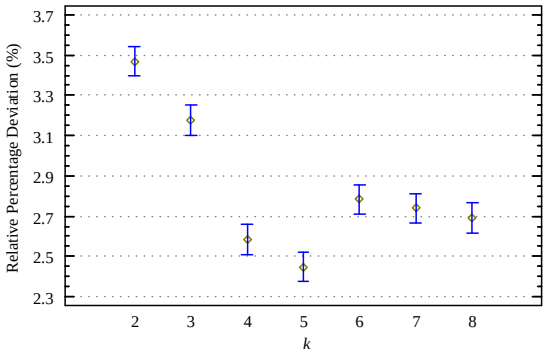

Figure 4: Means plot and 95% confidence level Tukey HSD intervals for various settings of parameter k on random instances with termination criterion set to $t = 20$.

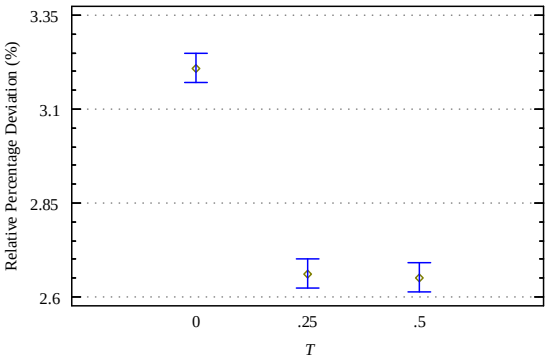

Figure 5: Means plot and 95% confidence level Tukey HSD intervals for various settings of parameter T on random instances with termination criterion set to $t = 20$.

some parameters in the two adapted algorithms. Therefore we calibrate the parameters k and T of the adapted IG_RS_{LS} and IGX for the coming comparisons. The interactions between the calibrated algorithms and the number of jobs destroyed k on calibration instances with termination criterion set to $t = 20$ are shown in Figure 6.

Figure 6 shows that there are no statistically significant differences for all k values, which is different from the permutation flowshop scheduling situation [RS07], where no statistically significant difference exists only for $k \in \{3,4,5\}$. We just take $k = 3$ in the following experiments for IG_RS_{LS}. In addition, there is no statistically significant difference for all $k \in \{2,3,4,5,6,7\}$ of the IGX [XZL12]. Since the lowest value occurs when $k = 5$, we set $k = 5$ in the following experiments for IGX. In concordance with the process used to obtain the results given by Ruiz and Stützle [RS07], we tested T of all the compared algorithms (IG_RS_{LS}, IGX and IG) using the above instances. We found that there is no statistically significant difference with all T for each of the three algorithms. This is similar to the results of IG shown in Figure 5 and also in line with the original experiments shown in [RS07]. Therefore, T takes 0.5 for all three algorithms in the following experiments.

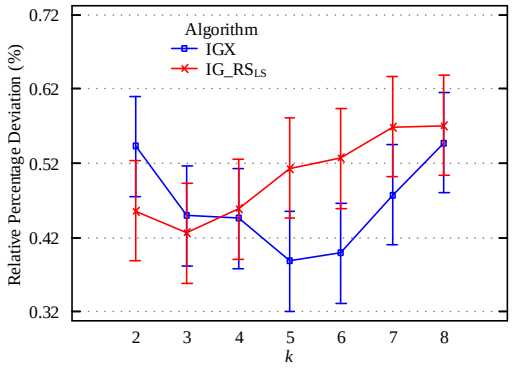

Figure 6: Interaction between the two calibrated existing algorithms and the number of jobs destroyed k. Analysis on random calibration instances with termination criterion set to $t = 20$.

6.1.3. Component Calibration

For each of the components of IG there are several variants. The initial solution can be constructed by Random (which constructs the initial solution randomly), the proposed ISC and NEH [NEH83]. The means plot and 95% Tukey HSD intervals of the three methods on the random calibration instances are shown in Figure 7. From Figure 7, it can be observed that the RPDs of the three initial solution construction methods show statistically significant differences. ISC has the lowest RPD. Though NEH is worse than ISC, it is much better than the Random construction.

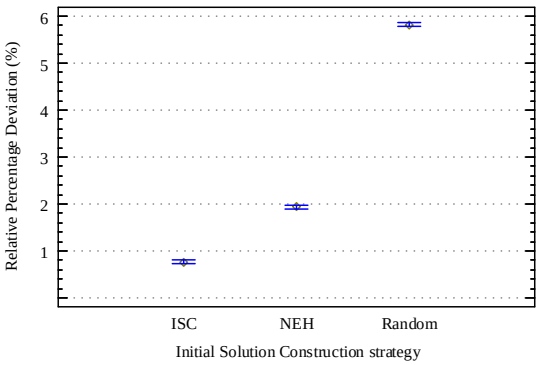

Figure 7: Means plot and 95% Tukey HSD intervals of the three initial solution construction methods on random calibration instances with termination criterion set to $t = 20$.

Since the local search of the IG_RS_{LS} of [RS07] is a commonly used local search, we consider three variants for the local search of IG: LS_6, IG_RS_{LS} and NoLS (no local search). The means plot and 95% confidence level Tukey HSD intervals for the three local searches on random instances with termination criterion set to $t = 20$ is depicted in Figure 8. The result implies that the differences between any pair of the three cases are significant. LS_6 has the lowest RPD. The RPD of LS_6 is much smaller than those of IG_RS_{LS} and $NoLS$, which indicates that LS_6 is effective for the considered problem. However, it is strange that

IG_RS_{LS} has the largest RPD, being even worse than the no local search case. The main reason lies in that the objective increment property cannot be applied to IG_RS_{LS} which leads to a very slow local search. Within the limited computational time, IG_RS_{LS} can carry out a few iterations and then it results in the largest RPD. Therefore, it is demonstrated that the speed-up formulas that we have presented for the total flowtime minimization in the sequence dependent setup times no-wait flowshop with learning and forgetting effects are playing a major role in the performance of the proposed local search.

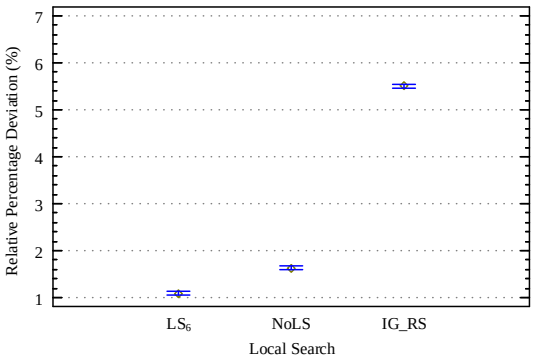

Figure 8: Means plot and 95% confidence level Tukey HSD intervals for the three local searches on random instances with termination criterion set to $t = 20$.

The Destruction & Reconstruction operator adopted by Ruiz and Stützle [RS07] is called DR in this paper, which is similar to MDR_0 but with different local researches and destruction & reconstruction processes. The three variants of MDR (MDR_0, MDR_1 and MDR_2) are compared with DR. The means plot and 95% Tukey HSD intervals of the four methods on the random calibration instances are shown in Figure 9. It can be observed that the RPD differences are statistically significant among the four operators. MDR_1 is significantly different from the other three processes. MDR_1 has the lowest RPD whereas MDR_2 has the largest. The fact that the RPD of MDR_2 is worse than that of MDR_1 demonstrates that worse accepted solutions can lead to better final solutions. The reason lies in that π would be worse than π_0 during the MDR. This is also true as the RPD of MDR_0 is worse than that of MDR_1. Therefore MDR_1 is utilized in the proposed IG in the following experiments.

6.2. Algorithm comparisons

According to the determined parameters and components, the proposed IG is evaluated by comparing it against the existing algorithms for similar scheduling problems. Since the considered problem has not been studied before there are no specific algorithms for it. We adapt some additional classical algorithms for similar problems. These are BIH [LPS99], TRIPS, TRIPS_M, QUARTS [NMA15], in addition to the already mentioned and calibrated IG_RS_{LS} [RS07] and IGX [XZL12]. BIH obtains a sequence of n jobs in n iterations. In each iteration, a sub-sequence is considered and the best sequence is obtained by inserting an unscheduled job into any position of the given sequence. TRIPS selects three jobs sequentially from the unscheduled job list. The first job of the best three job combination is removed

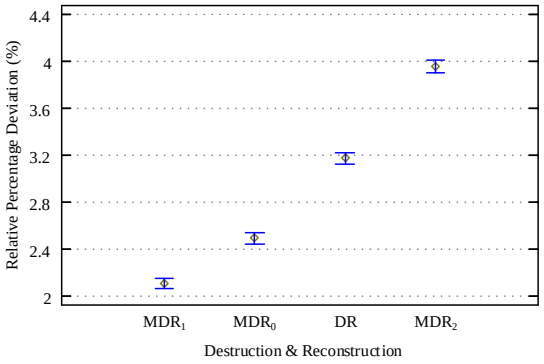

Figure 9: Means plot and 95% Tukey HSD intervals of the Destruction & Reconstruction methods on random instances with termination criterion set to $t = 20$.

and appended to the scheduled sub-sequence. The process is repeated until there are only three jobs remaining. The best sequence of the last three jobs is appended to the scheduled sub-sequence. QUARTS is similar to TRIPS but selects four jobs instead of three jobs every time and appends the first two jobs to the scheduled sub-sequence. The second job is set as the first job of the four jobs of the next iteration. The second job of the new best 4-job sequence is added to the scheduled sub-sequence. The process is repeated until there are only three jobs left and the best permutation of the last three jobs is appended to the scheduled sub-sequence. The obtained solution is improved by a neighbourhood insertion and permutation procedures. TRIPS_M improves the solution obtained by TRIPS using the neighbourhood insertion and permutation procedures adopted in QUARTS.

It has to be noted that BIH, TRIPS, TRIPS_M and QUARTS are one-pass deterministic heuristics and have a different termination criterion. The proposed IG, IG_RS_{LS} and IGX are nondeterministic algorithms and use the maximum CPU time $(n \times m \div 2) \times t$ $(t \in \{30, 60, 90\})$ milliseconds as the termination criterion. Therefore, the proposed IG is compared with the tested algorithms in different ways. All algorithms are compared on benchmark instance sets used in [RS08], i.e., SDST10, SDST50, SDST100 and SDST125 (the ratios of the setup times to the processing times are at most 10%, 50%, 100% and 125%, respectively, of the maximum possible processing times of Taillard original instances). The instances can be downloaded from http://soa.iti.es/.

6.2.1. Nondeterministic algorithms comparison

IG is compared with IG_RS_{LS} and IGX on the benchmark instances. k takes 5 in IG, 3 in IG_RS_{LS} and 5 in IGX as mentioned above. The comparison of the results of the three algorithms with $(n \times m \div 2) \times t$ $(t \in \{30, 60, 90\})$ milliseconds on the instance sets SDST10, SDST50, SDST100 and SDST125 are shown in Table 1. We use the average RPD (ARPD) on five replicates to measure the effectiveness of the compared algorithms.

Table 1 shows that the proposed IG outperforms IG_RS_{LS} and IGX for all instances with all three termination criteria. ARPDs of IG for all cases are 0. IG_RS_{LS} outperforms IGX for most $n = 20, 50$ cases. However, the ARPD of IG_RS_{LS} gets much worse than that of IGX as n increases. Therefore, IG_RS_{LS} shows higher ARPD than IGX on average. For

Table 1: ARPD of IG_RS_{LS}, IGX and the proposed IG with $(n \times m \div 2) \times t$ milliseconds CPU time stopping criterion (best values in bold).

t	$n \times m$	SDST10			SDST50			SDST100			SDST125		
		IG_RS_{LS}	IGX	IG	IG_RS_{LS}	IGX	IG	IG_RS_{LS}	IGX	IG	IG_RS_{LS}	IGX	IG
30	20×5	0.03	0.05	**0.00**	**0.00**	0.11	**0.00**	**0.00**	0.34	**0.00**	**0.00**	0.79	**0.00**
	20×10	**0.00**	0.11	**0.00**	**0.00**	0.11	**0.00**	**0.00**	0.10	**0.00**	**0.00**	0.13	**0.00**
	20×20	**0.00**	0.02	**0.00**	**0.00**	0.11	**0.01**	**0.00**	0.02	**0.00**	**0.00**	0.11	**0.00**
	50×5	1.18	0.88	**0.00**	1.25	1.36	**0.00**	2.36	2.10	**0.00**	2.97	2.24	**0.00**
	50×10	0.50	0.67	**0.00**	0.97	1.07	**0.00**	1.42	1.30	**0.00**	1.59	1.57	**0.01**
	50×20	0.37	0.81	**0.00**	0.54	1.15	**0.00**	0.77	1.29	**0.00**	1.06	1.22	**0.00**
	100×5	2.25	1.56	**0.00**	2.89	3.20	**0.00**	5.21	4.29	**0.00**	5.48	5.65	**0.00**
	100×10	1.59	1.65	**0.00**	2.23	2.00	**0.00**	2.82	2.66	**0.00**	3.57	3.18	**0.00**
	100×20	1.44	1.23	**0.00**	1.63	1.37	**0.00**	2.32	1.74	**0.00**	1.98	1.86	**0.00**
	200×10	10.87	2.76	**0.00**	10.37	3.13	**0.00**	11.95	4.34	**0.00**	11.43	4.82	**0.00**
	200×20	8.50	2.52	**0.00**	8.10	2.28	**0.00**	8.06	2.61	**0.00**	7.24	2.68	**0.00**
	500×20	7.88	2.30	**0.00**	7.52	2.68	**0.00**	6.76	3.03	**0.00**	6.34	3.06	**0.00**
	Average	2.88	1.21	**0.00**	2.96	1.55	**0.00**	3.47	1.99	**0.00**	3.47	2.28	**0.00**
60	20×5	**0.00**	0.13	**0.00**	0.01	0.14	**0.00**	**0.00**	0.18	**0.00**	**0.00**	0.18	**0.00**
	20×10	**0.00**	0.01	**0.00**	**0.00**	0.06	**0.00**	**0.00**	0.03	**0.00**	**0.00**	0.18	**0.00**
	20×20	**0.00**	0.02	**0.00**	**0.00**	0.09	**0.01**	**0.00**	0.03	**0.00**	**0.00**	0.19	**0.00**
	50×5	0.88	0.98	**0.00**	1.31	1.41	**0.00**	2.44	2.12	**0.00**	3.35	2.78	**0.00**
	50×10	0.62	0.70	**0.00**	0.88	0.77	**0.00**	1.42	1.12	**0.00**	1.64	1.49	**0.00**
	50×20	0.43	0.88	**0.00**	0.42	0.97	**0.00**	0.87	1.00	**0.00**	0.86	1.12	**0.00**
	100×5	2.27	1.66	**0.00**	3.16	2.77	**0.00**	5.32	4.16	**0.00**	5.89	4.91	**0.00**
	100×10	1.54	1.76	**0.00**	2.11	1.74	**0.00**	3.12	2.72	**0.00**	3.31	3.06	**0.00**
	100×20	1.37	1.33	**0.00**	1.68	1.50	**0.00**	1.98	1.96	**0.00**	2.12	1.88	**0.00**
	200×10	9.82	2.40	**0.00**	10.65	3.11	**0.00**	11.62	4.36	**0.00**	11.53	4.89	**0.00**
	200×20	9.00	2.41	**0.00**	8.50	2.29	**0.00**	7.94	2.76	**0.00**	7.65	2.80	**0.00**
	500×20	8.26	2.58	**0.00**	7.58	2.66	**0.00**	6.84	3.08	**0.00**	6.76	3.35	**0.00**
	Average	2.85	1.24	**0.00**	3.02	1.46	**0.00**	3.46	1.96	**0.00**	3.59	2.24	**0.00**
90	20×5	**0.00**	0.12	**0.00**	**0.00**	0.04	**0.00**	**0.00**	0.17	**0.00**	**0.00**	0.47	**0.00**
	20×10	**0.00**	0.24	**0.00**	**0.00**	0.10	**0.00**	**0.00**	0.01	**0.00**	**0.00**	0.16	**0.00**
	20×20	**0.00**	0.02	**0.00**	**0.00**	0.04	**0.01**	**0.00**	0.09	**0.00**	**0.00**	0.12	**0.00**
	50×5	1.16	0.80	**0.00**	1.15	1.13	**0.00**	2.38	1.67	**0.00**	2.81	2.59	**0.00**
	50×10	0.50	0.62	**0.00**	0.98	0.86	**0.00**	1.33	1.26	**0.00**	1.78	1.32	**0.00**
	50×20	0.38	0.53	**0.00**	0.46	0.84	**0.00**	0.82	1.12	**0.00**	0.88	0.95	**0.00**
	100×5	2.44	1.84	**0.00**	3.27	2.63	**0.00**	4.73	3.81	**0.00**	5.88	4.68	**0.00**
	100×10	1.73	1.61	**0.00**	2.15	1.70	**0.00**	3.02	2.73	**0.00**	3.15	2.76	**0.00**
	100×20	1.29	1.29	**0.00**	1.59	1.38	**0.00**	1.90	1.80	**0.00**	1.96	1.84	**0.00**
	200×10	10.68	2.52	**0.00**	10.74	3.32	**0.00**	11.50	4.13	**0.00**	12.45	4.73	**0.00**
	200×20	8.91	2.12	**0.00**	7.74	2.14	**0.00**	8.10	2.61	**0.00**	7.75	2.62	**0.00**
	500×20	8.25	2.68	**0.00**	7.83	2.80	**0.00**	7.10	3.24	**0.00**	7.04	3.68	**0.00**
	Average	2.95	1.20	**0.00**	2.99	1.42	**0.00**	3.41	1.89	**0.00**	3.64	2.16	**0.00**

example, the ARPDs of IG_RS_{LS} are 0 for the three $n = 20$ cases and $t = 30$ on the instance set SDST50. However, the ARPD of IG_RS_{LS} becomes 10.37% for the 200×10 case, which is much worse than that of IGX which is only 3.13%. The average ARPD of IG_RS_{LS} is 2.96% and that of IGX is 1.55%. In other words, the proposed IG is the most robust algorithm for the considered problem and IGX is more robust than IG_RS_{LS}. As the ratio of the setup times to the processing times increases, the ARPDs of both IG_RS_{LS} and IGX increase. For example, when $t = 30$, the average ARPDs of IG_RS_{LS} and IGX are 2.88% and 1.21% for the SDST10 set while they become 3.47% and 2.28% as the ratio increases to 125%, i.e.,

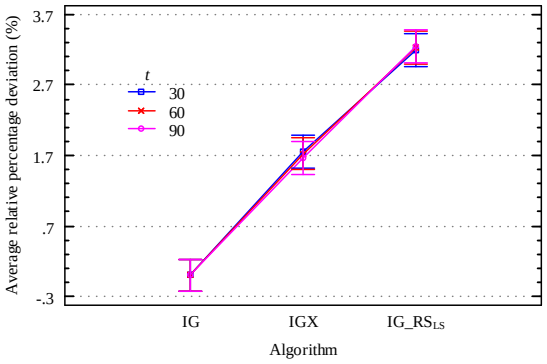

Figure 10: Interaction plot between the tested algorithms and the termination criterion $t \in \{30, 60, 90\}$ with 95% Tukey HSD intervals on instance set SDST125.

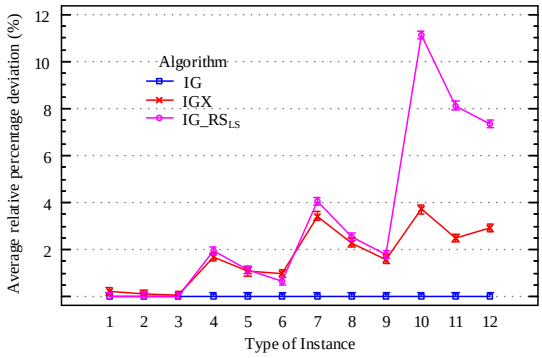

Figure 11: Interaction plot between the tested algorithms and the type of instances with 95% Tukey HSD intervals on instance set SDST125 with $t = 90$.

SDST125. However, the termination criterion has little influence on the performance of IG_RS_{LS} and IGX (which is also verified in Figure 10).

Though a large number of tests have been carried out, comparing algorithms on the basis of means is potentially misleading. The ANOVA technique is used to analyze the results in a sound and statistical way. Figure 10 shows the interaction plot between the tested algorithms and the termination criterion $t \in \{30, 60, 90\}$ with 95% Tukey HSD intervals on the instance set SDST125. Figure 11 shows the interaction plot between the tested algorithms and the type of instance with 95% Tukey HSD intervals on the instance set SDST125 with $t = 90$.

From Figure 10, we can observe that t has little impact on the performance of all three algorithms. In addition, IG_RS_{LS} is the worst among the three and IGX is worse than the proposed IG, which implies that IG is the most suitable for the considered problem. For a better insight into the performance of the three algorithms on the 12 instance groups (the 9 combinations of $n \in \{20, 50, 100\}$ and $m \in \{5, 10, 20\}$, 200×10, 200×20, and 500×20) of SDST125, Figure 11 illustrates that the proposed IG is the best in ARPD. Though the performance of IG_RS_{LS} and IGX is similar to that of the proposed IG when the type is less than 4, it becomes worse for the larger types. In addition, it can be observed that IG_RS_{LS} and IGX perform worse for the same n and smaller m. For example, the ARPD of IG_RS_{LS} is 12.45% when $t = 90$ for the 200×10 group on SDST125 while it is 7.75% for the 200×20 group; for the $n = 100$ groups, the ARPD of IG_RS_{LS} is 5.88% when $m = 5$ while those are 3.15% and 1.96% when $m = 10$ and $m = 20$ respectively.

6.2.2. Comparison with heuristics

The proposed IG is compared with BIH, TRIPS, TRIPS_M and QUARTS on the benchmark instances. Because the maximum iteration number N is the termination criterion for the deterministic heuristics, N is examined first. For this we revert again to the 340 calibration instances. The means plot of the ARPD and 95% Tukey HSD intervals for N is shown in Figure 12.

Figure 12 shows that there is no statistically significant difference in the ARPD for every

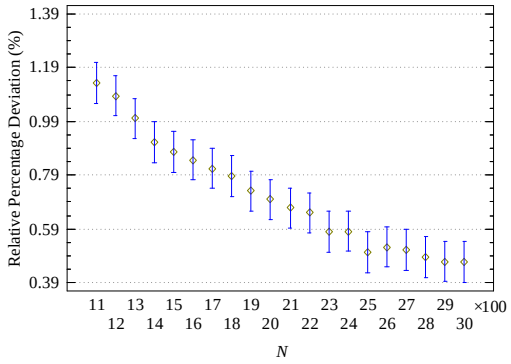

Figure 12: Means plot of ARPD and 95% Tukey HSD intervals for the iteration number N in the heuristics comparison.

100 iteration increase. For example, the average ARPD difference between $N = 1800$ and $N = 3000$ is less than 0.4%. It seems that there are few variations $N \geq 2800$. Therefore we take $N = 2900$ for the proposed IG in the following experiments. The other heuristics are one-pass methods. For the benchmark instances, the results of the compared heuristics are shown in Table 2.

Table 2 shows that the proposed IG has the smallest ARPD among the compared algorithms on the different instance sets, i.e., the proposed IG outperforms the other compared heuristics. The average ARPD of the proposed IG is 0. For the considered problem, BIH is the second best heuristic. QUARTS is slightly better than TRIPS_M. TRIPS is always the worst in ARPD and QUARTS is the most time-consuming. TRIPS_M and TRIPS have similar computation times for all instances, which are much less than that of QUARTS but much more than those of BIH and the proposed IG. Though BIH needs less computation time than the proposed IG when $n \leq 200$, it requires more CPU time than the proposed IG when $n = 500$. For example, QUARTS takes 11548.84s, TRIPS_M 1584.32s, TRIPS 1573.00s, BIH 165.64s, and the proposed IG only 91.90s for the 500×20 group of SDST10. Table 2 also shows that the ratio of the setup times to the processing times exerts little influence on both ARPD and CPU times for the compared methods, i.e., each of the methods have a similar average ARPD and CPU time on every instance set.

Again, the ANOVA technique is used to analyze the ARPD of all the methods on the benchmark instances and the ARPD of the methods on each group of the instance set SDST50. The means and interaction plots of ARPD with 95% Tukey HSD intervals are shown in Figures 13 and 14.

From Figure 13, it can be concluded that there are statistically significant differences among the compared methods, which is in accordance with Table 2. The proposed IG has the smallest ARPD. Figure 14 shows the interaction between the type of instances and algorithms on SDST50. It can be observed that the proposed IG is the most robust among the compared methods. Furthermore, the ARPDs of the other heuristics fluctuate with different instance groups though there are no statistically significant differences in most cases.

All experimental results are available on the website: www.seu.edu.cn/lxp/75/5b

Table 2: ARPD and CPU times (in seconds) for the proposed IG and the tested heuristics on SDST10, SDST50, SDST100 and SDST125 instances (best values in bold).

Set	$n \times m$	BIH		TRIPS		TRIPS_M		QUARTS		IG	
		ARPD	Time	ARPD	Time	ARPD	Time	ARPD	Time	ARPD	Time
SDST10	20×5	2.00	**0.00**	14.38	**0.00**	9.77	**0.00**	6.91	0.03	**0.00**	0.09
	20×10	2.42	**0.00**	9.50	**0.00**	6.14	**0.00**	5.07	0.01	**0.00**	0.04
	20×20	1.76	**0.00**	7.84	**0.00**	6.11	**0.00**	2.90	0.01	**0.00**	0.04
	50×5	3.48	**0.01**	15.29	0.06	12.45	0.06	10.07	0.49	**0.00**	0.21
	50×10	3.47	**0.01**	9.03	0.06	7.03	0.06	7.09	0.52	**0.00**	0.19
	50×20	3.06	**0.01**	7.69	0.06	6.21	0.06	5.29	0.56	**0.00**	0.19
	100×5	3.91	**0.06**	14.21	1.19	12.07	1.24	10.83	8.06	**0.00**	0.92
	100×10	3.98	**0.06**	12.17	1.28	11.17	1.23	7.80	7.83	**0.00**	1.15
	100×20	3.22	**0.06**	8.62	1.17	7.49	1.23	5.97	7.94	**0.00**	1.21
	200×10	3.51	**1.24**	10.43	15.65	9.75	14.94	8.73	151.99	**0.00**	5.24
	200×20	3.04	**1.14**	7.50	13.25	7.05	13.75	6.12	144.79	**0.00**	4.56
	500×20	2.41	165.64	7.41	1573.00	7.14	1584.32	6.81	11548.84	**0.00**	**91.90**
	Average	3.02	14.02	10.34	133.81	8.53	134.74	6.96	989.26	**0.00**	**8.81**
SDST50	20×5	3.06	**0.00**	11.96	**0.00**	8.22	**0.00**	5.47	0.03	**0.00**	0.08
	20×10	1.82	**0.00**	8.12	**0.00**	5.41	**0.00**	4.11	0.01	**0.00**	0.05
	20×20	1.80	**0.00**	6.56	**0.00**	4.52	**0.00**	3.28	0.01	**0.00**	0.06
	50×5	3.19	**0.00**	12.69	0.07	10.70	0.06	9.55	0.50	**0.00**	0.22
	50×10	3.05	**0.00**	8.40	0.07	7.20	0.07	8.20	0.60	**0.00**	0.18
	50×20	2.71	**0.01**	6.77	0.07	5.51	0.07	5.19	0.59	**0.00**	0.18
	100×5	3.88	**0.07**	12.94	1.31	11.47	1.40	9.35	10.23	**0.00**	0.97
	100×10	2.93	**0.07**	9.73	1.35	8.98	1.40	7.10	9.93	**0.00**	2.04
	100×20	2.72	**0.06**	7.53	1.26	6.68	1.35	5.17	10.17	**0.00**	1.39
	200×10	3.33	**1.37**	8.74	22.62	8.18	21.65	7.10	216.85	**0.00**	4.93
	200×20	2.99	**1.04**	6.56	15.12	6.09	15.72	5.40	162.75	**0.00**	4.27
	500×20	2.25	179.93	6.84	1638.26	6.64	1647.90	6.00	12318.98	**0.00**	**92.96**
	Average	2.81	15.21	8.90	140.01	7.47	140.80	6.33	1060.89	**0.00**	**8.94**
SDST100	20×5	3.14	**0.00**	12.83	0.01	8.57	**0.00**	6.52	0.03	**0.00**	0.11
	20×10	2.31	**0.00**	7.81	**0.00**	5.04	0.01	4.04	0.01	**0.00**	0.08
	20×20	1.68	**0.00**	5.38	**0.00**	3.58	**0.00**	3.13	0.01	**0.00**	0.11
	50×5	5.51	**0.00**	12.72	0.06	11.37	0.06	8.50	0.43	**0.00**	0.48
	50×10	3.51	**0.00**	8.19	0.07	6.95	0.07	6.16	0.56	**0.00**	0.54
	50×20	2.64	**0.01**	5.86	0.07	4.97	0.07	4.30	0.54	**0.00**	0.64
	100×5	5.73	**0.06**	14.67	1.28	13.41	1.31	10.75	9.89	**0.00**	2.25
	100×10	4.13	**0.06**	9.74	1.28	8.88	1.37	7.41	9.86	**0.00**	2.13
	100×20	3.13	**0.06**	6.72	1.23	6.13	1.32	4.99	9.61	**0.00**	1.78
	200×10	4.33	**1.04**	9.66	14.43	9.23	14.75	8.15	148.02	**0.00**	4.44
	200×20	2.99	**1.06**	6.30	13.73	5.93	14.90	5.59	156.48	**0.00**	4.23
	500×20	2.38	149.61	6.15	1576.40	5.99	1579.15	5.71	12130.56	**0.00**	**99.91**
	Average	3.46	12.66	8.84	134.05	7.50	134.42	6.27	1038.83	**0.00**	**9.73**
SDST125	20×5	3.41	**0.00**	10.35	**0.00**	7.38	**0.00**	4.81	0.03	**0.00**	0.08
	20×10	2.68	**0.00**	8.16	**0.00**	6.04	**0.00**	3.74	0.01	**0.00**	0.10
	20×20	1.88	**0.00**	5.03	**0.00**	3.69	**0.00**	2.91	0.01	**0.00**	0.11
	50×5	5.62	**0.00**	13.07	0.05	11.28	0.05	9.69	0.34	**0.00**	0.31
	50×10	3.73	**0.00**	8.55	0.04	7.41	0.04	6.88	0.28	**0.00**	0.48
	50×20	3.16	**0.00**	5.85	0.04	4.91	0.05	5.63	0.33	**0.00**	0.46
	100×5	5.60	**0.05**	14.92	0.86	13.67	0.86	10.07	7.58	**0.00**	2.00
	100×10	4.08	**0.05**	9.15	0.83	8.33	0.97	7.11	7.50	**0.00**	1.88
	100×20	3.36	**0.04**	6.64	0.76	6.03	0.81	5.39	7.06	**0.00**	1.70
	200×10	4.13	**1.15**	9.59	17.19	9.11	15.62	7.66	180.99	**0.00**	5.19
	200×20	3.36	**1.19**	6.10	16.75	5.78	15.58	5.37	187.64	**0.00**	4.36
	500×20	2.36	166.17	6.07	1652.67	5.91	1658.24	5.68	12587.16	**0.00**	**82.97**
	Average	3.61	14.06	8.62	140.77	7.46	141.02	6.24	1081.58	**0.00**	**8.30**

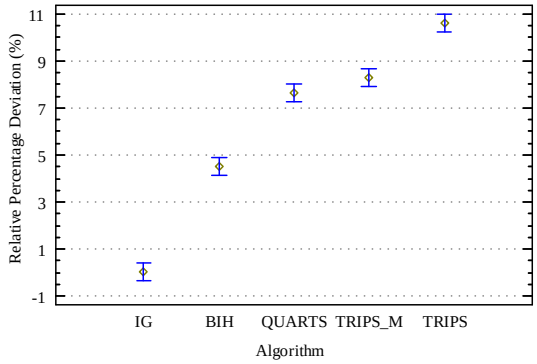

Figure 13: Means plot of ARPD and 95% Tukey HSD intervals of the five methods on the instance set SDST50.

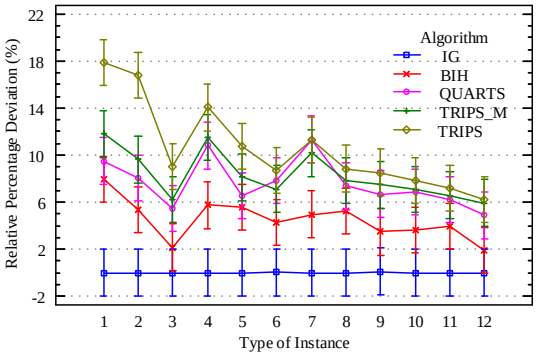

Figure 14: Interaction plot of ARPD and 95% Tukey HSD intervals of the five methods on the instance set SDST50.

/c12114a161115/page.psp.

7. Conclusions and Future work

In this paper, a learning and forgetting effects model is constructed for the sequence dependent setup time no-wait flowshop to minimize the total flowtime. To speed up the search process, objective incremental properties are deduced which are different from the traditional properties for no-wait flowshops without learning and forgetting effects. In terms of objective incremental properties, an initial solution construction method is developed which is shown to be more effective than existing methods. To enhance the intensification of the proposed algorithm, variable neighborhood descent (VND) methods are investigated, among which VND_6 is verified to be the most effective through comprehensive statistical experimentations. A modified destruction & reconstruction procedure is presented to improve diversification. An iterated greedy algorithm IG is proposed. This IG is compared with existing heuristics (BIH, TRIPS, TRIPS_M, QUARTS) and non-deterministic metaheuristic algorithms (IG_RS_{LS} and IGX) on four instance sets with different termination criteria. Experimental and statistical results show that the ARPD of the proposed IG is 0 most of the time which is much less than those of IG_RS_{LS} and IGX using the same CPU time. The proposed IG outperforms BIH, TRIPS, TRIPS_M, QUARTS in terms of effectiveness. TRIPS, TRIPS_M and QUARTS use much more computational time than the proposed IG. The running time of the proposed IG is less than 6 seconds for $n \leq 200$, which is desirable in practice.

Future avenues of research include other objectives in no-wait flowshops (e.g., total tardiness) which are common in some practical industries and more learning and forgetting factors for real-time applications. Other generalized problems with applications in industry as those shown by Pan et al. [PWM+13] or Li et al. [LP16] could be extended with the consideration of learning and forgetting effects. Applying learning and forgetting effects to the setup times themselves is also a very interesting problem that has been, to the best of our knowledge, ignored in the scheduling literature. It has to be stressed that setup times

are operations carried out in machines usually by trained personnel subject to learning and forgetting effects. Studying other metaheuristic approaches is always a worthwhile effort as more refined methods might be able to reach even better solutions.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Nos. 61572127, 61272377), the Key Research & Development program in Jiangsu Province (No. BE2015728), the Collaborative Innovation Center of Wireless Communications Technology and the Key Natural Science Fund for Colleges and Universities in Jiangsu Province (No. 12KJA630001). Rubén Ruiz is partially supported by the Spanish Ministry of Economy and Competitiveness, under the project "SCHEYARD - Optimization of Scheduling Problems in Container Yards" with reference DPI2015-65895-R.

[AJRA14] H Asefi, F Jolai, M Rabiee, and ME Tayebi Araghi. A hybrid nsga-ii and vns for solving a bi-objective no-wait flexible flowshop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, 75(5-8):1017–1033, 2014.

[AS13] Sedighe Arabameri and Nasser Salmasi. Minimization of weighted earliness and tardiness for no-wait sequence-dependent setup times flowshop scheduling problem. *Computers & Industrial Engineering*, 64(4):902–916, 2013.

[Bis99] Dirk Biskup. Single-machine scheduling with learning considerations. *European Journal of Operational Research*, 115(1):173–178, 1999.

[BR03] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308, 2003.

[BZ13] J. Behnamian and M. Zandieh. Earliness and tardiness minimizing on a realistic hybrid flowshop scheduling with learning effect by advanced metaheuristic. *Arabian Journal for Science and Engineering*, 38(5):1229–1242, 2013.

[CT12] Yu-Hsiang Chung and Lee-Ing Tong. Bi-criteria minimization for the permutation flowshop scheduling problem with machine-based learning effects. *Computers & Industrial Engineering*, 63(1):302–312, 2012.

[CW00] T.C.Edwin Cheng and Guoqing Wang. Single machine scheduling with learning effect considerations. *Annals of Operations Research*, 98(1-4):273–290, 2000.

[GPSL13] Kaizhou Gao, Quanke Pan, P.N. Suganthan, and Junqing Li. Effective heuristics for the no-wait flow shop scheduling problem with total flow time minimization. *International Journal of Advanced Manufacturing Technology*, 66(9-12):1563–1572, 2013.

[GS88] SK Goyal and C Sriskandarajah. No-wait shop scheduling: computational complexity and approximate algorithms. *Opsearch*, 25(4):220–244, 1988.

[HS96] N. G. Hall and C. Sriskandarajah. A Survey of Machine Scheduling Problems with Blocking and No-Wait in Process. *Operations Research*, 44(3):510–525, 1996.

[LC09] Dipak Laha and UdayK. Chakraborty. A constructive heuristic for minimizing makespan in no-wait flow shop scheduling. *The International Journal of Advanced Manufacturing Technology*, 41(1-2):97–109, 2009.

[LC13] Wen-Chiung Lee and Yu-Hsiang Chung. Permutation flowshop scheduling to minimize the total tardiness with learning effects. *International Journal of Production Economics*, 141(1):327–334, 2013.

[Lee04] Wen-Chiung Lee. A note on deteriorating jobs and learning in single-machine scheduling problems. *International Journal of Business and Economics*, 3(1):83–89, 2004.

[LP16] J.-Q. Li and K. Pan, Quan-Ke Mao. A hybrid fruit fly optimization algorithm for the realistic hybrid flowshop rescheduling problem in steelmaking systems. *IEEE Transactions on Automation Science and Engineering*, 13(2):932–949, 2016.

[LPS99] L.Bianco, P.Dell'Olmo, and S.Giordani. Flow shop no-wait scheduling with sequence-dependent setup times and release dates. *INFOR*, 37(1):3–19, 1999.

[LWS04] WC Lee, CC Wu, and HJ Sung. A bi-criterion single-machine scheduling problem with learning considerations. *ACTA INFORMATICA*, 20(4):303–315, 2004.

[LWW08] Xiaoping Li, Qian Wang, and Cheng Wu. Heuristic for no-wait flow shops with makespan minimization. *International Journal of Production Research*, 46(9):2519–2530, 2008.

[Mos01] Gur Mosheiov. Scheduling problems with a learning effect. *European Journal of Operational Research*, 132(3):687–693, 2001.

[NA14] Marcelo Seido Nagano and Daniella Castro Araújo. New heuristics for the no-wait flowshop with sequence-dependent setup times problem. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 36(1):139–151, 2014.

[NEH83] Muhammad Nawaz, E Emory Enscore, and Inyong Ham. A heuristic algorithm for the m-machine n-job flow-shop sequencing problem. *Omega*, 11(1):91–95, 1983.

[NM16] Marcelo Seido Nagano and Hugo Hissashi Miyata. Review and classification of constructive heuristics mechanisms for no-wait flow shop problem. *The International Journal of Advanced Manufacturing Technology*, 86(5-8):2161–2174, Jan 2016.

[NMA15] Marcelo Seido Nagano, Hugo Hissashi Miyata, and Daniella Castro Araújo. A constructive heuristic for total flowtime minimization in a no-wait flowshop with sequence-dependent setup times. *Journal of Manufacturing Systems*, 36:224–230, 2015.

[NZ14] B. Naderi and M. Zandieh. Modeling and scheduling no-wait open shop problems. *International Journal of Production Economics*, 158:256–266, 2014.

[PR12] Quan-Ke Pan and Rubén Ruiz. Local search methods for the flowshop scheduling problem with flowtime minimization. *European Journal of Operational Research*, 222(1):31–43, 2012.

[PR13] Quan-Ke Pan and Rubén Ruiz. A comprehensive review and evaluation of permutation flowshop heuristics to minimize flowtime. *Computers & Operations Research*, 40(1):117–128, 2013.

[PWM+13] Quan-Ke Pan, L. Wang, K. Mao, J.-H. Zhao, and M. Zhang. An effective artificial bee colony algorithm for a real-world hybrid flowshop problem in steelmaking process. *IEEE Transactions on Automation Science and Engineering*, 10(2):307–322, 2013.

[PWS+13] Quan-Ke Pan, L. Wang, H.-Y. Shang, J.-Q. Li, and M. Liu. A high performing memetic algorithm for the flowshop scheduling problem with blocking. *IEEE Transactions on Automation Science and Engineering*, 10(3):741–756, 2013.

[PWZ08] Quan-Ke Pan, L. Wang, and B.-H. Zhao. An improved iterated greedy algorithm for the no-wait flow shop scheduling problem with makespan criterion. *International Journal of Advanced Manufacturing Technology*, 38(7-8):778–786, 2008.

[PZ12] F. Pargar and M. Zandieh. Bi-criteria SDST hybrid flow shop scheduling with learning effect of setup times:water flow-like algorithm approach. *International Journal of Production Research*, 50(10):2609–2623, 2012.

[Raj93] Chandrasekharan Rajendran. Heuristic algorithm for scheduling in a flowshop to minimize total flowtime. *International Journal of Production Economics*, 29(1):65–73, 1993.

[RCTM11] I. Ribas, R. Companys, and X. Tort-Martorell. An iterated greedy algorithm for the flowshop scheduling problem with blocking. *Omega*, 39(3):293–301, 2011.

[RS07] Rubén Ruiz and Thomas Stützle. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177(3):2033–2049, 2007.

[RS08] Rubén Ruiz and Thomas Stützle. An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. *European Journal of Operational Research*, 187(3):1143–1159, 2008.

[RZ97] Chandrasekharan Rajendran and Hans Ziegler. An efficient heuristic for scheduling in a flowshop to minimize total weighted flowtime of jobs. *European Journal of Operational Research*, 103(1):129–138, 1997.

[TCW11] Sunantha Teyarachakul, Suresh Chand, and James Ward. Effect of learning and forgetting on

batch sizes. *Production and Operations Management*, 20(1):116–128, 2011.

[VNFR13] Behdin Vahedi Nouri, Parviz Fattahi, and Reza Ramezanian. Hybrid firefly-simulated annealing algorithm for the flow shop problem with learning effects and flexible maintenance activities. *International Journal of Production Research*, 51(12):3501–3515, 2013.

[Wan06] J.-B. Wang. A note on scheduling problems with learning effect and deteriorating jobs. *International Journal of Systems Science*, 37(12):827–833, 2006.

[WC07a] Ji-Bo Wang and TC Edwin Cheng. Scheduling problems with the effects of deterioration and learning. *Asia-Pacific Journal of Operational Research*, 24(2):245–261, 2007.

[WC07b] Xiuli Wang and T.C. Edwin Cheng. Single-machine scheduling with deteriorating jobs and learning effects to minimize the makespan. *European Journal of Operational Research*, 178(1):57–70, 2007.

[WJCW12] Ji-Bo Wang, P. Ji, T.C.E. Cheng, and Dan Wang. Minimizing makespan in a two-machine flow shop with effects of deterioration and learning. *Optimization Letters*, 6(7):1393–1409, 2012.

[Wri36] T. P Wright. Factors affecting the cost of airplanes. *Journal of the Aeronautical Sciences*, 3(4):122–128, 1936.

[WW14] Ji-Bo Wang and Jian-Jun Wang. Flowshop scheduling with a general exponential learning effect. *Computers & Operations Research*, 43(1):292–308, 2014.

[WZZ+13] Xiao-Yuan Wang, Zhili Zhou, Xi Zhang, Ping Ji, and Ji-Bo Wang. Several flow shop scheduling problems with truncated position-based learning effect. *Computers & Operations Research*, 40(12):2906–2929, 2013.

[XZL12] Tao Xu, Xia Zhu, and Xiaoping Li. Efficient iterated greedy algorithm to minimize makespan for the no-wait flowshop with sequence dependent setup times. In *Computer Supported Cooperative Work in Design (CSCWD), 2012 IEEE 16th International Conference on*, pages 780–785, May 2012.

[YC08] Wen-Hua Yang and Suresh Chand. Learning and forgetting effects on a group scheduling problem. *European Journal of Operational Research*, 187(3):1033–1044, 2008.

[YLHZ12] Yunqiang Yin, Min Liu, Jinghua Hao, and MengChu Zhou. Single-machine scheduling with job-position-dependent learning and time-dependent deterioration. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 42(1):192–200, Jan 2012.