4OR (2024) 22:31–45
https://doi.org/10.1007/s10288-023-00535-7

RESEARCH PAPER

An efficient iterated greedy algorithm for a multi-objective no-wait flow shop problem with sequence dependent setup times

Fernando Siqueira de Almeida[1] · Marcelo Seido Nagano[1]

Received: 13 January 2022 / Revised: 6 January 2023 / Accepted: 1 February 2023 /
Published online: 10 March 2023

Abstract
In this article, we address the m-machine no-wait flow shop scheduling problem with sequence dependent setup times. The objective is to minimize total tardiness subject to an upper bound on makespan. Although these constraints and performance measures have all been extensively studied independently, they have never been considered together in this problem before. So, this multi-criteria approach provides more realistic solutions for complex scenarios. To solve the problem, we developed a new heuristic called IG_A. The proposed method repeatedly performs a process of destruction and construction of an existing solution in order to improve it. The novelty of this method includes a mechanism capable of adapting the destruction intensity according to the instance size and the number of iterations, calibrating the algorithm during the search. Computational experiments indicate that IG_A outperforms the best literature method for similar applications in overall solution quality by about 35%. Therefore, IG_A is recommended to solve the problem.

Keywords Iterated greedy · Flow shop · No-wait · Sequence-dependent setup · Total tardiness · Makespan

Mathematics Subject Classification 68T20 · 90C59

Fernando Siqueira de Almeida and Marcelo Seido Nagano have contributed equally to this work.

✉ Marcelo Seido Nagano
drnagano@usp.br

Fernando Siqueira de Almeida
fernando.siqueira.almeida@gmail.com

[1] Department of Production Engineering, University of Sao Paulo, Av. Trabalhador Saocarlense, Sao Carlos, Sao Paulo 13566-590, Brazil

1 Introduction

Sequencing and scheduling can be defined as allocating resources to perform tasks over time. This type of problem is so general that it appears almost everywhere. Unfortunately, proposing feasible solutions in real-world situations is not always easy. In complex environments, scheduling problems often involve tasks competing for resources along with several embedded constraints and objectives, making them very hard to solve (Pinedo 2016; T'kindt and Billaut 2006). Furthermore, the pressure to overcome this challenge is ever-increasing, since failing to use available resources efficiently or respond quickly to customer demands in today's competitive markets can be the end of any business. This makes a well-chosen scheduling strategy play a crucial role in present times. In this context, we study the *m*-machine flow shop scheduling problem (FSP). In order to propose more realistic solutions for complex scenarios, the problem is addressed with a multiple criteria approach. Specifically, a multi-objective function is applied to minimize total tardiness subject to an upper bound on makespan. In addition, the FSP is considered with no-wait (NWT) and sequence dependent setup times (SDST). These objectives and constraints are common in productions systems and have never been considered together in this problem before. To solve the problem, a new algorithm is proposed and compared with the current best methods for similar applications.

The *m*-machine FSP is a system where a set of jobs have to be processed through a series of machines in the same processing routing. Each machine processes one job at a time. No job can overtake other jobs (*first-in-first-out*) and therefore only permutation schedules are allowed. When the NWT constraint is in place, there is no waiting time between successive operations. Therefore, no job is permitted to utilize a buffer or to wait in an upstream machine. NWT may occur due to process requirements or unavailability of waiting space. Having setup constraints means that a machine requires some preparation before processing a particular job. This preparation may include cleaning, retooling, adjustments, inspection and rearrangement of the workstation. When setup times are sequence-dependent, the length of these times depends on the difficulty involved in switching from one processing configuration to another. SDST are common in multipurpose machines or where a single facility produces a variety of products. In those situations, instead of absorbing the setup times in the processing times, it is recommended to make explicit considerations to address the problem (Baker and Trietsch 2019; Pinedo 2016; Emmons and Vairaktarakis 2013).

Total tardiness and makespan are two important performance measures in the field of scheduling. Total tardiness is the sum of all delays related to not meeting due dates. Minimizing total tardiness implies that delays can be subjected to penalties, but there are no benefits from completing jobs in advance. Makespan represents the maximal completion time among all jobs in the system. Minimizing makespan is appropriate when a complete batch of jobs needs to be dispatched as quickly as possible. A reduced makespan is also important to have an efficient utilization of resources, that is, to decrease equipment idle time (Baker and Trietsch 2019; Pinedo 2016). Total tardiness and makespan are addressed here as the hierarchical objective $TT \mid C_{max} \leq K$. A hierarchical function $A \mid B \leq K$ minimizes A subject to an upper bound K on B. This formulation is appropriate when there is no need to optimize objective B as long as it does not exceed the upper bound (Emmons and Vairaktarakis 2013).

The NWT-FSP is proved to be NP-*hard*. Since using exact methods in this case can be impractical, heuristic methods have been developed to speed up the process of finding reasonable solutions for this problem. Some researchers have proposed algorithms to minimize makespan and total tardiness in NWT-FSP-SDST. The most relevant studies include greedy algorithms (Bianco et al. 1999; Xu et al. 2012; Li et al. 2018), simulated annealing (Lee and Jung 2005; Aldowaisan and Allahverdi 2015), hybrid genetic algorithm (Franca et al. 2006), constructive heuristics (Ara and Nagano 2011), greedy randomized adaptive search procedure and evolutionary local search based (Zhu et al. 2013b), iterative algorithms (Zhu et al. 2013a), hybrid evolutionary cluster search (Nagano and Araújo 2014), hybrid greedy algorithm (Zhuang et al. 2014), particle swarm optimization (Samarghandi and ElMekkawy 2014), genetic algorithms (Samarghandi 2015a, b; Aldowaisan and Allahverdi 2015) and local search (Miyata et al. 2019).

The literature on multi-objective optimization of NWT-FSP is relatively limited, especially when hierarchical objectives are addressed. Allahverdi (2004) proposed an insertion-based iterative algorithm to minimize a weighted sum of makespan and maximum tardiness with an upper bound on maximum tardiness. Framinan and Leisten (2006) proposed a heuristic based on a dominance property over the job of maximum tardiness to minimize makespan subject to maximum tardiness. To minimize makespan subject to mean completion time (or the equivalent total completion time), Aydilek and Allahverdi (2012) presented several heuristics based on simulated annealing and local searches and Nagano et al. (2020) proposed an iterated greedy. Allahverdi and Aydilek (2013) develop an iterated local search and a genetic algorithm to minimize total completion time subject to makespan, and Allahverdi and Aydilek (2014) proposed—among others—a heuristic based on simulated annealing for the same problem with separate setup times. Recently, Allahverdi et al. (2018) proposed an algorithm which combines simulated annealing and an insertion algorithm to solve the NWT-FSP with total tardiness subject to makespan, and Allahverdi et al. (2020) presented a similar method for the same problem with separate setup times.

As can be noted, many applications and research have independently investigated different variations of the no-wait flow shop scheduling problem. And since most of them have not been compared, it is unclear which solution methodology performs better and under what conditions [see the related surveys presented by for more information]. However, the literature highlights the most frequently used high-performance techniques. Based on this information, we propose a new iterated greedy algorithm called IG_A. An iterated greedy algorithm fundamentally has a construction mechanism and a destruction mechanism that are repeatedly executed in an alternating order. It is a very flexible and malleable method that can easily be combined with other techniques and reach very high performance when applied to basically any combinatorial optimization problem. The iterated greedy was initially proposed to solve the FSP by Ruiz and Stützle (2007). After that, different extensions have been published considering additional constraints, e.g., sequence-dependent setup times (Ruiz and Stützle 2008), no-idle mixed no-idle (Pan and Ruiz 2014), no-wait (Yamada et al. 2021) and blocking (Tasgetiren et al. 2017). Other optimization criteria have also been addressed apart from makespan such as total tardiness and total flowtime. The iterated greedy has also been effective to solve multi-objective scheduling problems (Dubois-Lacoste et al. 2011; Minella et al. 2011; Ciavotta et al. 2013). Given the wide applicability, the flexibility, and the often high performance of iterated greedy, we decided to use this strategy to develop IG_A.

In this work, the no-wait flow shop problem to minimize total tardiness subject to makespan is addressed for the first time with sequence-dependent setup times, which is a more complex scenario in production systems where machine setup times depend on the sequencing of jobs and decisions taken in previous periods. We present an original iterated greedy algorithm. The novelty of this method includes a new mechanism capable of using some search-dependent properties to calibrate the algorithm during execution. Specifically, the algorithm adjusts its destruction intensity according to the instance size and number of iterations to improve convergence to a global optimum. Extensive computational experiments were conducted to evaluate the proposed approach against two recent methods for similar problems: algorithm AA (no setup) by Allahverdi et al. (2018) and algorithm PA (simple setup) by Allahverdi et al. (2020). The two literature methods are adapted to solve the problem with sequence-dependent setup times. The results show that the proposed algorithm outperforms the best literature method by about 35% on average under the same computational conditions.

The remaining content is structured as follows. Section 2 is dedicated to the problem definition. The algorithms are described in Sect. 3. Section 4 presents the computational experiments. The final conclusions and some future directions are given in Sect. 5.

2 Problem definition

The FSP consists of a set $M = \{1, \dots, m\}$ of m work stations that process a set $N = \{1, \dots, n\}$ of n independent jobs sequentially. In addition, there are some assumptions made on jobs and machines: all operational requirements are known; all jobs are available to be processed from time zero; there is only one machine on each work station (m-machine); no machine can process two or more jobs at the same time; no job can be processed on more than one machine simultaneously; each operation must be performed without interruption (no preempt); after an operation is completed, a job is immediately available for the next operation; each machine must process a job only once (no recirculation); all processing times are positive; and all jobs are processed in the same order on all machines (permutation). The goal consists of finding the best possible sequence that optimizes the objective function.

The problem can be addressed as follows. Let $p_{i,j}$ denote the processing time of job $j \in N$ on machine $i \in M$. Furthermore, let $s_{i,j}$ represent the setup time required on machine i for job j after job $j-1$ is completed. Then, the start time of job j on the first machine, denoted by S_j, can be defined as

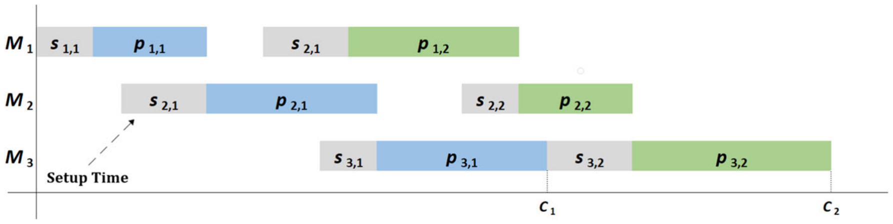


Fig. 1 Two adjacent jobs on the NWT-FSP-SDST

$$S_j = \max_{1<k\leq m} \left\{ S_{j-1} + \sum_{i=1}^{k} p_{i,j-1} + s_{k,j} - \sum_{i=1}^{k-1} p_{i,j} \right\}, \quad j = 2, \ldots, n$$

where $S_1 = s_{1,1}$ and $j = 0$ is a dummy job with processing times and setup times of zero length. Now, let C_j denote the completion time of job j on the last machine. Then, C_j can be defined as

$$C_j = S_j + \sum_{i=1}^{m} p_{i,j} \ , \quad j = 1, \ldots, n.$$

In addition, let TT and C_{max} denote total tardiness and makespan, respectively. Finally, let d_j be the due date of job j. Total tardiness is defined as $TT = \sum_{j=1}^{n} T_j$, where $T_j = max\{C_j - d_j; 0\}$. Makespan is defined as $C_{max} = C_n$. Figure 1 illustrates two adjacent jobs on the NWT-FSP-SDST.

Using the classification system $\alpha/\beta/\gamma$ introduced by the m-machine NWT-FSP-SDST to minimize total tardiness subject to makespan can be written as

$$Fm \ / \ nwt, s_{i,j} \ / \ (TT \mid C_{max} \leq K) \, ,$$

where Fm and nwt represent flow shop and no-wait, respectively. Thus, the problem consists of finding the best possible sequence so that total tardiness is minimized and makespan does not exceed a predefined K value. In real problems, the initial solution and K must be provided by the scheduler. For the experiments performed here, these data are generated using the Transposition local search presented by Algorithm 1. Starting from a random sequence, the Transposition method returns the best solution of $(n - 1)$ permutations produced by swapping successive pairs of adjacent jobs. The upper bound K is defined as the makespan value of the returned sequence.

Algorithm 1 Transposition

```
1: $\pi$ := random sequence;
2: $K$ := $C_{max}(\pi)$;
3: for $i$ := 1 to ($n$ - 1) do
4:     Insert the $i$-th job of $\pi$ to position $i + 1$;
5:     if $C_{max}(\pi) < K$ then
6:         $K$ := $C_{max}(\pi)$;
7:     else
8:         Move the $(i + 1)$-th job of $\pi$ back to position $i$;
9:     end if
10: end for
11: return $K, \pi$;
```

3 Heuristic algorithms

The problem of $Fm/nwt/$ (TT | $C_{max} \leq K$) has already been addressed in the literature by Allahverdi et al. (2018, 2020). In this work, the algorithm presented in each study is implemented to solve the problem with sequence-dependent setup times. The two methods are briefly explained in the next subsection, followed by a complete description of the proposed algorithm.

3.1 Literature algorithms

Allahverdi et al. (2018) proposed the algorithm *AA* to solve the NWT-FSP with the objective of minimizing total tardiness subject to makespan. They adapt many existing algorithms for this problem and prove that *AA* performs better than all of them. In summary, the algorithm *AA* is composed by a simulating annealing (SA) algorithm and an insertion local search. Starting from an initial sequence obtained from EDD (Earliest Due Date) rule, the SA is performed to reduce TT using a random pairwise exchange operator. Then, if the solution does not satisfy the constraint $C_{max} \leq K$, the insertion local search tries to improve the sequence to obtain a feasible solution. The algorithm repeats these steps until the stopping criterion is satisfied.

Allahverdi et al. (2020) studied the same problem with separate setup times. These setups can be classified as simple because the preparation times do not depend on production in previous periods. They propose the algorithm *PA*, which was shown to outperform different existing algorithms modified for this environment. *PA* starts the initial solution as the best sequence between the EDD rule and the NEH minimizing TT. The optimization process has three phases. In the first phase, an SA algorithm utilizes block insertion and block exchange operators to reduce TT. In the second phase, an insertion local search is performed as an attempt to find a feasible solution for the condition $C_{max} \leq K$. In the third phase, total tardiness is minimized by inserting the job having the maximum tardiness into all other positions.

It should be noted that at the start of both, *AA* and *PA*, the optimization of total tardiness takes place without imposing the upper bound on makespan. Only then does a local search try to satisfy this constraint, but this time without accepting worse values for total tardiness. For this reason, these algorithms tend to get stuck, as optimizing total tardiness and makespan are often competing objectives. In other words, both methods may not be able to generate a feasible solution. This fact is confirmed in the experiments carried out. To avoid fundamental changes in the algorithms, the initial solution is returned each time *AA* or *PA* are not able to produce a feasible solution.

3.2 Iterated greedy algorithm (IG_A)

The proposed method, called IG_A, optimizes an initial solution through a destruction and construction process. In the destruction phase, a set of random jobs is removed from the incumbent solution. This simple stochastic method provides fast destruction and at the same time reduces the chance of trapping in local optima. The construction phase tries to reinsert the removed jobs through a process based on the NEH mechanism, Nawaz et al. (1983) to produce a new candidate solution. Next, an acceptance criterion updates the incumbent with the candidate if the new sequence is a better feasible solution (or in other words, if the candidate is complete, has lowest total tardiness and satisfies the upper bound constraint on makespan). The method is composed of Algorithms 2 and 3.

Algorithm 2 IG_A

```
1: Initialize: N, P, $\pi_0$;
2: $\pi := \pi_0$;
3: $F := TT(\pi)$;
4: $f := 1 / exp(ln(2) \times (n/N)^P)$
5: $d := f$;
6: repeat
7:     $\pi' :=$ DestroyRepair($\pi, d$);
8:     $F' := TT(\pi')$;
9:     if $F' < F$ then
10:        $\pi := \pi'$;
11:        $F = F'$;
12:    end if
13:    $d = d \times f$;
14: until Time Limit
15: return $\pi$;
```

Algorithm 3 DestroyRepair

```
1: Initialize: $\pi, d$;
2: $\pi_d := \pi$;
3: $D := n \times d$;   # integer
4: $\pi_e := D$ random jobs removed from $\pi_d$
5: while $\pi_e$ is not empty do
6:     $\pi_d :=$ best feasible sequence generated by inserting the last job of $\pi_e$ in all positions of $\pi_d$
7: end while
8: if $\pi_d$ is feasible then
9:     $\pi := \pi_d$
10: end if
11: return $\pi$;
```

Algorithm 2 runs at the highest level. In line 1, the necessary parameters (N and P) and the initial sequence (s_0) are defined. The next two lines initialize the incumbent variables π and F that receive copies of s_0 and its total tardiness $TT(s_0)$, respectively. In line 4, the factor $f \in \{0, 1\}$ is calculated using N and P. A copy of f is made to initialize the variable d in line 5. Lines 6–14 make up the main loop, which is responsible for the optimization itself. First, Algorithm 3 is called on line 7. It takes π and d as arguments and returns the candidate solution π'. Then, an acceptance process is carried out from line 8–12. In this process, the auxiliary variable F' is initialized with $TT(\pi')$ in line 8. If in line 9 the statement $F' < F$ is true, that is, $TT(\pi') < TT(\pi)$, then π and F are updated with π' and F' in lines 10 and 11, respectively. After the acceptance process, the factor f is used to update d in line 13. If the time limit is not reached in line 14, the main loop starts another iteration from line 6. Otherwise, the iterations terminate and the incumbent sequence π is returned in line 15.

Algorithm 3 runs at the lowest level. It generates a candidate solution through the destruction and construction process. In line 1, the sequence π and the coefficient d are both initialized. In line 2, a copy of π is made to initialize the auxiliary variable π_d. The number of jobs to be removed from π_d is calculated and assigned to D in line 3. In line 4, the sequence π_e is initialized with D random jobs removed from π_d. In lines 5–7, a loop tries to reinsert the removed jobs, one by one, from π_e to π_d through a NEH-based process where the job to be inserted is tested in all possible positions. The best feasible solution is taken at the end of each iteration. These steps are repeated until all jobs are reinserted or no feasible solution is produced. When the reinsertion process can generate a feasible solution, π is updated with π_d (line 9). Finally, π is returned in line 11.

It is important to note that d is not fixed across different instance sizes or while the algorithm is solving a problem. Instead, some search-dependent properties are used by the algorithm to modify this value in order to find an appropriate balance between search diversification and intensification. Diversification is explored by removing a large number of jobs to drive the search to rather distant solutions in the construction phase. Intensification, on the other hand, is explored by removing few jobs to focus the search on more localized regions. Some initial experiments suggested that a diversification strategy is more effective at the beginning of the search and have the cost of higher computation times for large problems. Considering this observation, the algorithm was designed so that the initial value of d is inversely proportional to the number of jobs and it decreases after each iteration. The adjustment is made at the end of each loop by multiplying d by the constant f defined as

$$f = \frac{1}{exp(ln(2) \times (n/P)^{Q})},$$

where n is the number of jobs. The values P and Q are parameters that have to be calibrated. The parameter P defines the number of jobs needed for f to reach 50%. Q is the parameter used to attenuate the change rate of the function. Figure 2 illustrates the function with some values set for P and Q.

The IRACE software package (Lopez-Ibanez et al. 2016) is used to determine the appropriate parameter settings. This package automatically finds the most appropriate

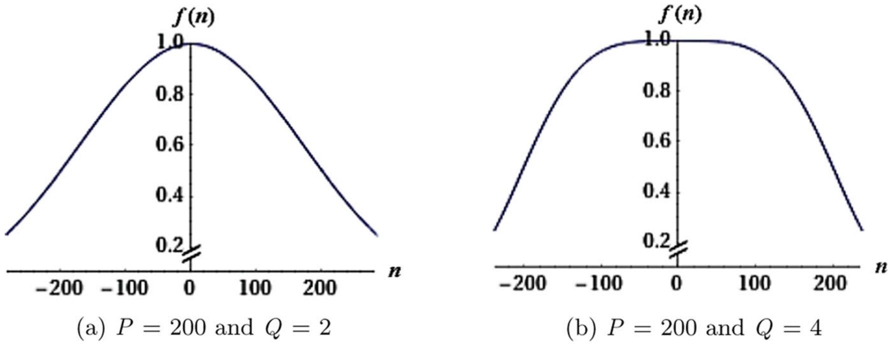
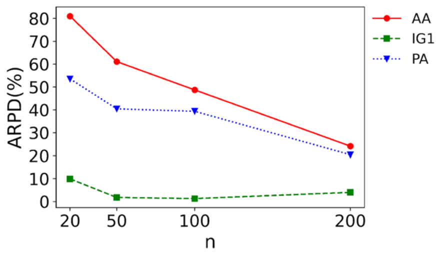


Fig. 2 Examples of function $f(n)$

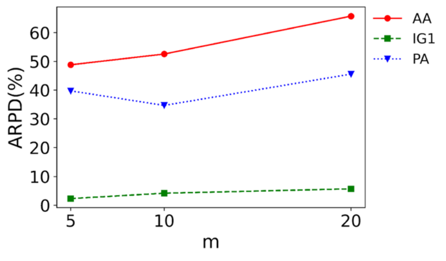
parameter values for optimization algorithms given a set of instances and parameter ranges. A set of 120 instances was created for this purpose, composed by all combinations of $n \in \{5, 10, 20, 40, 80, 160\}$ and $m \in \{4, 8, 12, 16\}$, with 5 different problems for each combination $n \times m$. It was considered for calibration $N \in \{150, 300, 450, 600, 750, 900\}$ and $P \in \{4^{-1}, 3^{-1}, 2^{-1}, 1, 2, 3, 4\}$. The processing times have a uniform distribution in the range [1, 99]. The sequence dependent setup times are at most 10% of the maximum processing times. In order to generate due dates, a uniform distribution between $L(1 - T - R/2)$ and $L(1 - T + R/2)$ was used where T represents a tardiness factor, R represents the due date range, and L denotes an approximate value for makespan. This is a consolidate approach in literature to generate due dates (e.g., Aldowaisan and Allahverdi 2012; Fernandez-Viagas and Framinan 2015; Framinan and Perez-Gonzalez 2018). The values 0.25 and 1.0 were defined for T and R, respectively. The tuning was performed using a computation time limit of $n \times (m/2) \times 25$ ms as stopping criterion (see Ruiz and Stützle 2008). The best parameter values obtained are $P = 300$ and $Q = 2$.

4 Computational experiments

All algorithms were implemented in C++. The computer used was a PC with an AMD Quad-Core Processor A12-9720P 3.60 GHz and 8 GB RAM running under a Windows 10 operating system. This study considers the test problems proposed by Minella et al. (2008) and Ruiz and Stützle (2008), which are extensions of the benchmarks of Taillard (1993). The processing times have a uniform distribution in the range [1, 99]. We use five instance sets with ratio of setup times $s \in \{0, 10, 50, 100, 125\}$ of the maximum processing times. Each set consists of 10 problems for each combination $n \times m$ of $\{20, 50, 100\} \times \{5, 10, 20\}$ and $200 \times \{10, 20\}$. The first set called SDST0 is taken from Minella et al. (2008) and has no setup times. The remaining sets called SDST10, SDST50, SDST100, and SDST125 are taken from Ruiz and Stützle (2008) and have times uniformly distributed in the range [1, 9], [1, 49], [1, 99] and [1, 124], respectively. Thus, there are 550 instances in total. In order to provide an easy and fair comparison,

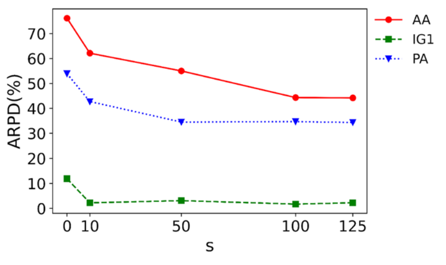


(a) $ARPD \times$ Jobs [n]

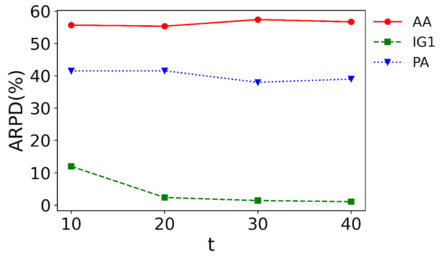


(b) $ARPD \times$ Machines [m]

Fig. 3 ARPD versus jobs and machines



(a) ARPD × Setup [s]



(b) ARPD × Time [t]

Fig. 4 ARPD versus setup and time

all methods were tested by using the same initial solutions (and K values) obtained from Algorithm 1.

The stopping criterion used was the CPU time limit defined as $n \times (m/2) \times t$ ms. The experiments were performed for $t \in \{10, 20, 30, 40\}$ to analyze the algorithms in different available computational times. The solutions were evaluated by using the average relative percentage deviation ($ARPD$) defined as

$$ARPD_h = \frac{100}{N} \sum_{i=1}^{N} \frac{TT_i^h - TT_i^{best}}{TT_i^h},$$

where $ARPD_h$ represents the performance of a heuristic h. In other words, this measure consists of the arithmetic mean of the deviations of a heuristic h from each best known solution. Therefore, the best heuristic is the one with the lowest ARPD value.

The ARPD results over jobs and machines are presented in Table 1(a). Table 1(b) gives the results over setup and time factors. It should be noted that the ARPD values of IG_A are disproportionately higher for time factor $t = 10$. The reason for this is that the time limits generated in this condition may not be sufficient for the algorithm to complete the first iteration, especially in cases with small instances. Others minor differences can be explained by the fact that the benchmarks used have due dates produced by different strategies. Despite that, the performance behaviors and their differences are very clear as can be seen in Figs. 3 and 4. As noted, the differences

Table 1 ARPD results

(a) Jobs (n) × machines (m)

Heuristic		IG_A	AA	PA
n	m			
20	5	3.75	55.51	48.86
	10	6.76	82.07	37.66
	20	19.76	106.97	75.39
50	5	1.83	47.65	34.93
	10	1.88	58.04	41.17
	20	1.60	77.56	45.15
100	5	1.39	43.25	35.39
	10	1.38	47.37	40.31
	20	1.08	55.55	42.34
200	10	6.75	22.77	19.50
	20	1.24	25.71	21.24
Average		4.23	56.28	39.98

(b) Setup (s) × time (t)

Heuristic		IG_A	AA	PA
s	t			
0	10	39.93	85.79	61.70
	20	4.87	75.48	57.11
	30	2.03	72.14	48.54
	40	0.85	71.28	48.36
10	10	4.65	58.63	41.67
	20	1.92	58.88	35.77
	30	1.09	69.62	40.70
	40	1.30	61.37	52.78
50	10	9.34	51.11	40.25
	20	1.56	55.93	31.90
	30	0.86	55.99	32.50
	40	0.75	57.02	33.41
100	10	3.33	43.51	32.19
	20	1.54	42.86	40.96
	30	1.20	45.01	38.86
	40	0.08	45.95	26.64
125	10	3.32	39.99	31.84
	20	2.08	43.95	42.17
	30	1.89	44.62	29.36
	40	1.72	48.18	34.05
Average		4.23	56.28	39.98

Table 2 Multiple comparison of means—Tukey' HSD test

Methods		Mean difference (I–J)	Std. error	Sig.	95% confidence interval		Reject
I	J				Lower bound	Upper bound	
AA	IG_A	52.06	1.28	0.000	49.04	55.07	True
	PA	16.30	1.28	0.000	13.29	19.31	True
IG_A	AA	– 52.06	1.28	0.000	– 55.07	– 49.04	True
	PA	– 35.76	1.28	0.000	– 38.77	– 32.74	True
PA	AA	– 16.30	1.28	0.000	– 19.31	– 13.29	True
	IG_A	35.75	1.28	0.000	32.74	38.77	True

FWER = 0.05

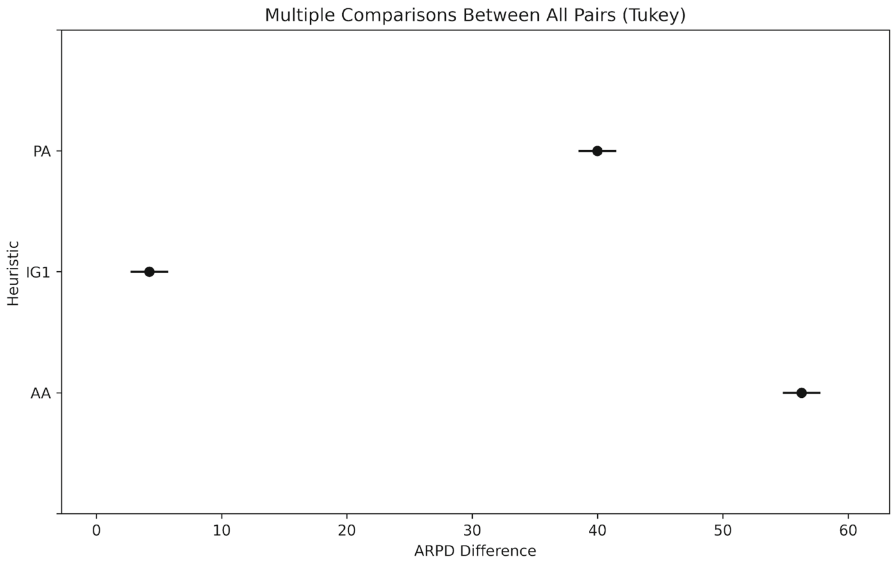


Fig. 5 Multiple comparisons between all pairs (Tukey)

between the algorithms tend to decrease when the setup factor and especially the number of jobs increase. On the other hand, in general, the errors get bigger when the number of machines increases. Variations in the time factor do not appear to have much effect for values above 10. In all cases, IG_A has a significant advantage over the other methods. The overall *ARPD* values of IG_A, *PA*, and *AA* are 4.23, 39.98 and 56.28, respectively.

The Tukey' honestly significant difference (HSD) test was conducted to analyze the statistic difference between the methods. The null hypothesis that two algorithms have equal performances was tested at a significance level of 5%. The results in Table 2 show that all algorithms are statistically different from each other. The differences at a confidence interval of 95% are illustrated in Fig. 5. As can be noted, IG_A outperform the second best method by about 35% and the third best by about 52%.

5 Conclusion

In this work, the NWT-FSP-SDST is addressed. The objective is to minimize total tardiness subject to the constraint that makespan does not exceed a maximum acceptable value. An iterated greedy algorithm was proposed to solve the problem. The proposed approach was tested against the existent algorithms *AA* and *PA*, which are designed to solve the most similar problems found in the literature. Experiments under the same computational conditions show that IG_A, *PA* and *AA* obtained overall *ARPD* values of 4.23, 39.98 and 56.28, respectively. Therefore, IG_A outperforms the literature methods.

There are some issues that can be investigated in the future. First, the proposed algorithm was compared with the best literature methods for similar problems. However, adapting other methods for different applications in future experiments could be promising. Another option is to consider additional factors like number of machines to determine the destruction strength of the proposed algorithm. In this work, only the number of jobs was considered for this purpose and therefore it may be possible to get better results with a broader approach. Additionally, the diversification/intensification behavior of the heuristic can be further explored. For example, it can be favorable to examine new strategies that combine greedy searching with random perturbations, both in destruction and construction phases.

Author Contributions Conceptualization, methodology, investigation, and original draft preparation FSA and MSN; review and editing, MSN and FSA; supervision, MSN; project administration and funding acquisition, FSA and MSN.

Funding Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq)—Brazil under Grant Numbers 306075/2017-2, 430137/2018-4 and 312585/2021-7. Coordenação de Aperfeiçoamento de Pessoal de Nível Superior—Brazil under Grant Number 88882.379108/2019-01.

Declarations

Conflict of interest The authors declare no conflict of interest.

References

Aldowaisan T, Allahverdi A (2012) Minimizing total tardiness in no-wait flowshops. Found Comput Decis Sci 37(3):149–162

Aldowaisan T, Allahverdi A (2015) Total tardiness performance in m-machine no-wait flowshops with separate setup times. Intell Control Autom 6:38–44

Allahverdi A (2004) A new heuristic for m-machine flowshop scheduling problem with bicriteria of makespan and maximum tardiness. Comput Oper Res 31(2):157–180

Allahverdi A, Aydilek H (2013) Algorithms for no-wait flowshops with total completion time subject to makespan. Int J Adv Manuf Technol 68(9–12):2237–2251

Allahverdi A, Aydilek H (2014) Total completion time with makespan constraint in no-wait flowshops with setup times. Eur J Oper Res 238(3):724–734

Allahverdi A, Aydilek H, Aydilek A (2018) No-wait flowshop scheduling problem with two criteria; total tardiness and makespan. Eur J Oper Res 269(2):590–601

Allahverdi A, Aydilek H, Aydilek A (2020) No-wait flowshop scheduling problem with separate setup times to minimize total tardiness subject to makespan. Appl Math Comput 365(124):688

Ara DC, Nagano MS (2011) A new effective heuristic method for the no-wait flowshop with sequence-dependent setup times problem. Int J Ind Eng Comput 2(1):155–166

Aydilek H, Allahverdi A (2012) Heuristics for no-wait flowshops with makespan subject to mean completion time. Appl Math Comput 219(1):351–359

Baker KR, Trietsch D (2019) Principles of sequencing and scheduling. Wiley, Hoboken

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

Ciavotta M, Minella G, Ruiz R (2013) Multi-objective sequence dependent setup times permutation flowshop: a new algorithm and a comprehensive study. Eur J Oper Res 227(2):301–313. https://doi.org/10.1016/j.ejor.2012.12.031

Dubois-Lacoste J, Lopez-Ibanez M, Stutzle T (2011) A hybrid tp plus pls algorithm for bi-objective flowshop scheduling problems. Comput Oper Res 38(8):1219–1236. https://doi.org/10.1016/j.cor.2010.10.008

Emmons H, Vairaktarakis G (2013) Flow shop scheduling. Theoretical results, algorithms, and applications. Springer, Berlin

Fernandez-Viagas V, Framinan JM (2015) Neh-based heuristics for the permutation flowshop scheduling problem to minimise total tardiness. Comput Oper Res 60:27–36

Framinan JM, Leisten R (2006) A heuristic for scheduling a permutation flowshop with makespan objective subject to maximum tardiness. Int J Prod Econ 99(1–2):28–40

Framinan JM, Perez-Gonzalez P (2018) Order scheduling with tardiness objective: improved approximate solutions. Eur J Oper Res 266(3):840–850

Franca PM, Tin G Jr, Buriol L (2006) Genetic algorithms for the no-wait flowshop sequencing problem with time restrictions. Int J Prod Res 44(5):939–957

Lee YH, Jung JW (2005) New heuristics for no-wait flowshop scheduling with precedence constraints and sequence dependent setup time. In: International conference on computational science and its applications. Springer, pp 467–476

Li X, Yang Z, Ruiz R et al (2018) An iterated greedy heuristic for no-wait flow shops with sequence dependent setup times, learning and forgetting effects. Inf Sci 453:408–425

Lopez-Ibanez M, Dubois-Lacoste J, Caceres LP et al (2016) The irace package: iterated racing for automatic algorithm configuration. Oper Res Perspect 3:43–58. https://doi.org/10.1016/j.orp.2016.09.002

Minella G, Ruiz R, Ciavotta M (2008) A review and evaluation of multiobjective algorithms for the flowshop scheduling problem. INFORMS J Comput 20(3):451–471

Minella G, Ruiz R, Ciavotta M (2011) Restarted iterated Pareto greedy algorithm for multi-objective flowshop scheduling problems. Comput Oper Res 38(11):1521–1533. https://doi.org/10.1016/j.cor.2011.01.010

Miyata HH, Nagano MS, Gupta JN (2019) Integrating preventive maintenance activities to the no-wait flow shop scheduling problem with dependent-sequence setup times and makespan minimization. Comput Ind Eng 135:79–104

Nagano MS, Araújo DC (2014) New heuristics for the no-wait flowshop with sequence-dependent setup times problem. J Braz Soc Mech Sci Eng 36(1):139–151

Nagano MS, Almeida FS, Miyata HH (2020) An iterated greedy algorithm for the no-wait flowshop scheduling problem to minimize makespan subject to total completion time. Eng Optim 53:1–19

Nawaz M, Enscore EE, Ham I (1983) A heuristic algorithm for the m-machine, n-job flowshop sequencing problem. Omega Int J Manag Sci 11(1):91–95. https://doi.org/10.1016/0305-0483(83)90088-9

Pan QK, Ruiz R (2014) An effective iterated greedy algorithm for the mixed no-idle permutation flowshop scheduling problem. Omega Int J Manag Sci 44:41–50. https://doi.org/10.1016/j.omega.2013.10.002

Pinedo M (2016) Scheduling. Theory, algorithms, and systems, 5th edn. Springer, Berlin

Ruiz R, Stützle T (2007) A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. Eur J Oper Res 177(3):2033–2049

Ruiz R, Stützle T (2008) An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. Eur J Oper Res 187(3):1143–1159

Samarghandi H (2015a) A no-wait flow shop system with sequence dependent setup times and server constraints. IFAC PapersOnLine 48(3):1604–1609

Samarghandi H (2015b) Studying the effect of server side-constraints on the makespan of the no-wait flow-shop problem with sequence-dependent set-up times. Int J Prod Res 53(9):2652–2673

Samarghandi H, ElMekkawy TY (2014) Solving the no-wait flow-shop problem with sequence-dependent set-up times. Int J Comput Integr Manuf 27(3):213–228

Taillard E (1993) Benchmarks for basic scheduling problems. Eur J Oper Res 64(2):278–285. https://doi.org/10.1016/0377-2217(93)90182-M

Tasgetiren MF, Kizilay D, Pan QK et al (2017) Iterated greedy algorithms for the blocking flowshop scheduling problem with makespan criterion. Comput Oper Res 77:111–126. https://doi.org/10.1016/j.cor.2016.07.002

Tkindt V, Billaut JC (2006) Multicriteria scheduling: theory, models and algorithms. Springer, Berlin

Xu T, Zhu X, Li X (2012) Efficient iterated greedy algorithm to minimize makespan for the no-wait flowshop with sequence dependent setup times. In: Proceedings of the 2012 IEEE 16th international conference on computer supported cooperative work in design (CSCWD). IEEE, pp 780–785

Yamada T, Nagano M, Miyata H (2021) Minimization of total tardiness in no-wait flowshop production systems with preventive maintenance. Int J Ind Eng Comput 12(4):415–426

Zhu X, Li X, Gupta JN (2013a) Iterative algorithms for no-wait flowshop problems with sequence-dependent setup times. In: 2013 25th Chinese control and decision conference (CCDC). IEEE, pp 1252–1257

Zhu X, Li X, Wang Q (2013b) An adaptive intelligent method for manufacturing process optimization in steelworks. In: Proceedings of the 2013 IEEE 17th international conference on computer supported cooperative work in design (CSCWD). IEEE, pp 363–368

Zhuang WJ, Xu T, Sun MY (2014) A hybrid iterated greedy algorithm for no-wait flowshop with sequence dependent setup times to minimize makespan. In: Advanced materials research. Trans Tech Publ, pp 459–466