Engineering Optimization

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/geno20

# A novel iterated greedy algorithm for no-wait permutation flowshop scheduling to minimize weighted quadratic tardiness

Bruno de Athayde Prata & Marcelo Seido Nagano

**To cite this article:** Bruno de Athayde Prata & Marcelo Seido Nagano (2022): A novel iterated greedy algorithm for no-wait permutation flowshop scheduling to minimize weighted quadratic tardiness, Engineering Optimization, DOI: 10.1080/0305215X.2022.2144274

**To link to this article:** https://doi.org/10.1080/0305215X.2022.2144274

Published online: 24 Nov 2022.

ENGINEERING OPTIMIZATION
https://doi.org/10.1080/0305215X.2022.2144274



# A novel iterated greedy algorithm for no-wait permutation flowshop scheduling to minimize weighted quadratic tardiness

Bruno de Athayde Prata [a] and Marcelo Seido Nagano [b]

[a]Department of Industrial Engineering, Federal University of Ceara, Fortaleza, Brazil; [b]Department of Production Engineering, University of São Paulo, São Carlos, Brazil

**ABSTRACT**

This article addresses the no-wait permutation flowshop layout where the jobs must be processed continuously in the available machines without interruption. The objective function is weighted quadratic tardiness minimization. Since the problem under study is NP-hard, an iterated greedy algorithm is introduced where the parameters are selected according to a variable neighbourhood descent framework. In the authors' innovative algorithm, in the first iteration of the search process, the parameters are initialized with a fixed number. In the remaining iterations, such parameters are reduced deterministically, aiming to explore distinct values throughout the search process. The proposal is compared with five other algorithms proposed for closely related problems, considering two performance measures: the Relative Deviation Index (RDI) and the Success Rate (SR). The proposed algorithm produced average values of RDI and SR of 1.2% and 86.9%, respectively. The computational results demonstrated that the authors' proposal outperformed all the other five algorithms under comparison.

**ARTICLE HISTORY**
Received 22 February 2022
Accepted 7 September 2022

**KEYWORDS**
Production scheduling; weighted quadratic tardiness; no-wait flowshop; iterated greedy algorithm

## 1. Introduction

Given its several industrial applications, the no-wait flowshop scheduling problem has been widely studied in the last few years. Commonly, the no-wait constraint appears in distinct real-world scenarios, such as in plastic, chemical and pharmaceutical production Allahverdi (2016). In the no-wait flowshop environment, after the beginning of operations, $n$ jobs must be processed continuously on $m$ machines without preemption or interruption.

In the available literature on no-wait flowshops, most contributions are related to makespan (Ding *et al.* "An Improved Iterated Greedy Algorithm," 2015; Zhao *et al.* 2018, 2019; Miyata, Seido Nagano, and Gupta 2019; Nagano, de Almeida, and Miyata 2021) or total completion time (Ye *et al.* 2017; Della Croce, Grosso, and Salassa 2021) performance measures. Nevertheless, in several practical contexts, the consideration of problem due dates is more relevant to the greater satisfaction of customer needs. Usually, total tardiness minimization is an interesting performance measure that considers due dates. Nevertheless, a linearization of the tardiness measure can lead to distortions in the planning process.

Although linear tardiness considers all the jobs, this performance measure fails to evaluate the distribution of tardiness among the jobs. In a real-world scenario, tardiness distribution may be very important for planners. Large tardiness values are significantly penalized by weighted quadratic tardiness, avoiding schedules where the majority of the tardiness is centred on one or a few jobs. Several researchers have reported the importance of quadratic tardiness measures in production

**CONTACT** B. de A. Prata baprata@ufc.br

scheduling problems (Luh and Hoitomt 1993; Sun, Noble, and Klein 1999). Some studies have also reported contributions for single-machine (Thomalla 2001; Schaller and Valente 2012; Valente and Schaller 2012; Gonçalves, Valente, and Schaller 2016), parallel-machines (Hoitomt *et al.* 1990; Schaller and Valente 2018) and flowshop (Costa, Valente, and Schaller 2020; Silva, Valente, and Schaller 2022) environments.

Consider an illustrative example with five jobs wherein a given feasible solution for each job presents a tardiness equal to one unit of time. Another solution occurs when four jobs meet the established due dates, and a single job has tardiness equal to five units of time. It can be observed that both solutions present the same total tardiness value; however, they are distinct in managerial aspects. In the first solution, there is an equilibrium in order fulfilment. On the other hand, in the second solution, one of the customers is highly penalized. Linear total tardiness minimization is not able to reflect such situations. Thereby, the consideration of weighted quadratic tardiness is of great practical significance since this assumption potentially reflects more precisely the needs of the customers.

A new variant of the permutation flowshop scheduling problem is introduced with the no-wait constraint and weighted quadratic tardiness minimization. The main contributions of the present article are listed as follows.

- A Mixed-Integer Linear Programming (MILP) formulation for the variant under study is proposed. This model is based on positional decision variables with a quadratic objective function.
- An Iterated Greedy (IG) algorithm with a random destruction mechanism – called a Random Destruction Iterated Greedy (RDIG) algorithm – is proposed.
- An IG metaheuristic with dynamic selection of its parameters – called a Variable Iterated Greedy Descent (VIGD) algorithm – is proposed.
- Extensive computational experimentation with a set of 800 test instances proposed by Ruiz and Allahverdi (2009) is presented. Computational results pointed to the superiority of the present authors' proposed VIGD metaheuristic in comparison with several other solution approaches.

Here, the next sections of this article are described: in Section 2, the problem background is presented. In Section 3, the proposed MILP formulation for the problem is presented, while in Section 4 the proposed solution approaches are described. In Section 5, the results of computational experiments are presented and discussed, and finally, in Section 6, some conclusions and suggestions for future works are addressed.

## 2. Related literature

Several studies have reported results related to the no-wait flowshop scheduling problem (Allahverdi 2016). Nevertheless, studies addressing the weighted quadratic tardiness objective are quite limited in the available literature. Thereby, some related approaches to the problem under study are presented.

Costa, Valente, and Schaller (2020) presented the first study on the permutation flowshop to minimize the weighted squared tardiness. Several dispatch rules, as well as improvement procedures, were proposed. Computational results demonstrated that the algorithms developed for quadratic tardiness outperformed the algorithms available for the linear case. Considering this same environment, Silva, Valente, and Schaller (2022) presented an MILP model and four metaheuristics as solution approaches. The four considered metaheuristics were Iterated Local Search (ILS), IG, Variable Greedy (VG), and Steady-State Genetic Algorithms (SSGAs). Computational results showed that the IG metaheuristic presented the best results in comparison with the other evaluated metaheuristics.

Concerning the no-wait flowshop with tardiness-related objectives, the following contributions are summarized. Ruiz and Allahverdi (2009) studied the no-wait flowshop problem minimizing a weighted sum of makespan and maximum lateness. Several solution procedures were presented, such as dominance relations, constructive algorithms and metaheuristics. Aldowaisan and

Allahverdi (2012) addressed the $m$-machine no-wait scheduling problem for total tardiness minimization. Four metaheuristics were developed as solution approaches. The computational experience highlighted that the Further Improved Simulated Annealing (FISA) algorithm was the best. Even considering the no-wait flowshop with total tardiness minimization, Ding *et al.*"Accelerated Methods for Total Tardiness," (2015) proposed an acceleration procedure for the NEH algorithm (Nawaz, Enscore Jr, and Ham 1983). Two IG metaheuristics were proposed with this ANEH algorithm as the initial solution. Aldowaisan and Allahverdi (2015) introduced the no-wait flowshop scheduling problem with sequence-independent setup times with total tardiness minimization. Four metaheuristics were proposed, and the Further Improved Genetic Algorithm (FIGA2) presented the best results. Allahverdi, Aydilek, and Aydilek (2020) considered the no-wait flowshop scheduling problem with sequence-independent setup times with an upper bound to the makespan. A block simulated annealing metaheuristic was developed as the solution procedure. Lin, Lu, and Ying (2018) addressed the no-wait flowshop minimizing the sum of makespan and total weighted tardiness. A novel metaheuristic called the Cloud Theory-based Iterated Greedy (CTIG) metaheuristic was proposed.

Concerning the literature review, the following research gaps can be emphasized. A no-wait flowshop to minimize the weighted quadratic tardiness was not reported, despite its practical and theoretical importance. Additionally, the no-wait flowshop to minimize total tardiness has not been sufficiently studied. Thus, there are still opportunities for the proposal of efficient solution procedures, such as metaheuristics.

## 3. Mathematical formulation

Let $n$ be the number of jobs to be produced on a set of $m$ machines. For all the considered machines, a permutation $\pi$ represents a feasible sequence in which a given job $j$ can be processed in position $k$ of this permutation. Each job $j$ presents a processing time $p_{ij}$, a weight $w_j$ and a due date $d_j$. A job scheduled in position $k$ of the sequence presents a completion time $C_{ki}$ on machine $i$ as well as a tardiness $T_k$, which can be determined by the following expression: $T_k = \max\{C_{km} - d_k, 0\}$. In addition, the no-wait assumption is considered where the processing operations of the jobs on the available machines cannot be interrupted. Given these definitions, the problem under study is to find the best sequence to minimize the total weighted quadratic tardiness for all the jobs. A mixed-integer linear programming formutation with binary positional decision variables is proposed to control the allocation of jobs in the production sequence (Stafford Jr and Tseng 2002; Stafford Jr, Tseng, and Gupta 2005). The notation used hereinafter for the proposed model is presented as follows.

**Indices**

$i$: index for machines $\{1,2,\ldots,m\}$.
$j$: index for jobs $\{1,2,\ldots,n\}$.
$k$: index for positions $\{1,2,\ldots,n\}$.

**Parameters**

$p_{ji}$: processing time of job $j$ on machine $i$.
$w_j$: weight of job $j$.
$d_j$: due date of job $j$.

**Decision variables**

$T_k$: tardiness of job in position $k$.
$C_{ki}$: completion time of job in position $k$ on machine $i$.

$$x_{jk} = \begin{cases} 1, & \text{if job } j \text{ is in sequence position } k \\ 0, & \text{otherwise.} \end{cases}$$

The resulting MILP model is as follows.

minimize

$$\sum_{k=1}^{n} w_k T_k^2 \tag{1}$$

subjectto

$$T_k \geq C_{km} - \sum_{j=1}^{n} d_j x_{jk}, \quad \forall_k \tag{2}$$

$$C_{1,1} = \sum_{j=1}^{n} p_{j1} x_{j1} \tag{3}$$

$$C_{ki} = C_{k,i-1} + \sum_{j=1}^{n} p_{ji} x_{jk}, \quad \forall_{k,i>1} \tag{4}$$

$$C_{ki} \geq C_{k-1,i} + \sum_{j=1}^{n} p_{ji} x_{jk}, \quad \forall_{k>1,i} \tag{5}$$

$$\sum_{k=1}^{n} x_{jk} = 1, \quad \forall_j \tag{6}$$

$$\sum_{j=1}^{n} x_{jk} = 1, \quad \forall_k \tag{7}$$

$$T_k \geq 0, \quad \forall_k \tag{8}$$

$$C_{ik} \geq 0, \quad \forall_{i,k} \tag{9}$$

$$x_{jk} \in \{0, 1\}, \quad \forall_{j,k}. \tag{10}$$

The objective function (1) is the weighted squared tardiness minimization. Constraint set (2) calculates the tardiness for each job processed in position $k$. If binary decision variable $X_{jk}$ is equal to one, the job $j$ is scheduled in position $k$ and the due date $j$ is selected. Constraint (3) enforces that the completion time of the first job on the first machine is equal to the processing time of the job in the first sequence position on machine 1. Constraints (4) and (5) ensure that the completion and starting times are consistent with a no-wait flowshop environment. Constraint set (6) forces that a given job is processed in a single position of the sequence. Constraint set (7) guarantees that a given position in the sequence can receive one job. Finally, constraint sets (8), (9) and (10) establish the scope of the decision variables.

## 4. Proposed solution approaches

### 4.1. Preliminary remarks

The IG was proposed by Ruiz and Stützle (2007) for the permutation flowshop to minimize makespan. In essence, the IG is composed of two stages: the first one is a destruction phase, in which a given permutation is destroyed, and the second one is a construction phase, in which the destroyed solution is repaired. Usually, the new permutation is accepted using a probabilistic criterion (based on an SA algorithm). Both stages are repeated considering some stop criterion, such as the number of iterations or time limit.

The destruction procedure plays a key role in the performance of the IG algorithm. In this sense, the parameter $Q$ controls the number of jobs to be removed from the current permutation. Usually, this parameter receives a fixed value during the number of iterations. In the view of the present authors, before several preliminary computational experiments, the consideration of a fixed $Q$ value presents some disadvantages, listed as follows.

- The consideration of a fixed value for $Q$ implies a parameter to be calibrated.
- Throughout the search process, different values for $Q$ can improve the solutions found. For example, in the first iterations, higher values for $Q$ can contribute to the diversification of the search, and smaller values for Q can result in the intensification of the search in the last iterations.
- By means of variable values for $Q$, the IG can present a higher possibility of escaping from local optima.

In the following subsections, two proposals are presented that aim to mitigate such disadvantages. In the first algorithm, the parameter $Q$ is selected from a given list in each iteration. In the second one, the parameters of the IG algorithm are dynamically calibrated throughout the search process.

### 4.2. Random destruction iterated greedy (RDIG) algorithm

Based on the above considerations, a simple variant of the IG algorithm is proposed in which a value for the destruction parameter from a list is randomly selected at each iteration of the search process. Although this strategy does not consider the evaluated values of the objective function and the number of iterations, its robustness could be verified based on preliminary computational experiments.

At the begining of the search, an initial solution is generated using the well-known Earliest Due Date (EDD) dispatch rule (Jackson 1955). This initial schedule is then improved with the NEH algorithm. During a given time limit, the following operators are executed. First, a value for $Q$ is selected from a $Q_{\text{values}}$ list. Subsequently, the current schedule is destroyed, and a new permutation is constructed based on the NEH algorithm. Thereafter, a 2-opt local search procedure is applied with the best improvement strategy. Through this strategy, the entire neighbourhood is investigated; therefore, the best objective value of this portion of the search space is returned. Figure 1 describes an illustrative example of the local search. Consider a test instance with ten jobs and three machines. The initial solution is given by the permutation $\pi = \{2, 1, 3, 4, 5, 6, 7, 8, 9, 10\}$. The first movement of the local search generates a new solution $\pi = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ which reduces the tardiness of several jobs. After that, the acceptance criterion is applied. Finally, the generated solution is accepted according to the simulated annealing acceptance criterion. Better solutions are always accepted, and worse solutions are accepted according to the following probability:

$$R \leq \exp(f(\pi') - f(\pi))/T, \tag{11}$$

where $R$ is a random number between zero and one, $\pi'$ is the generated solution, and $\pi$ is the current solution. If the current time is equal to the specified time limit, the algorithm is finished and the best solution found is returned. Algorithm 1 describes the pseudocode of the proposed RDIG algorithm.

**Data:** $p$, $d$, $Q_{\text{values}}$, $T$
**Result:** A schedule $\pi$
Generate an initial solution
Apply NEH algorithm
**while** *current_time* $\leq$ *time_limit* **do**
    Randomly select the parameter $Q_v$ from the $Q_{\text{values}}$ list
    Apply destruction operator
    Apply construction operator
    Apply local search
    Acceptance criterion
**end**

**Algorithm 1:** RDIG algorithm.

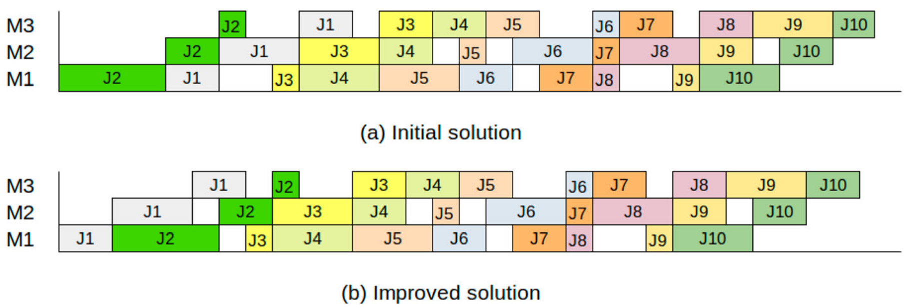


**Figure 1.** An illustrative example of the local research approach.

### 4.3. Variable iterated greedy descent

In the RDIG algorithm, a fixed temperature value is adopted. Additionally, the number of iterations is not considered to select the size of the destruction in the current permutation. Thus, a novel IG metaheuristic is proposed considering a dynamic variation of both parameters.

The Variable Neighbourhood Search (VNS) algorithm applies several neighbourhoods in a randomized way. The Variable Neighbourhood Descent (VND) metaheuristic deterministically explores the selected neighbourhoods (Mladenović and Hansen 1997; Hansen and Mladenović 2001). In this context, the concept of VND is used to variate the parameters of the IG throughout the search process. A linear decreasing of the parameters is considered, as suggested by Muller, Spoorendonk, and Pisinger (2012). In each iteration of the search process, the parameters $Q$ and $T$ are determined using the Equations (12) and (13), respectively:

$$Q = \left\lceil Q_{\text{start}} - (Q_{\text{start}} - Q_{\text{end}}) \times \frac{t_{\text{current}}}{t_{\text{limit}}} \right\rceil \quad (12)$$

$$T = \left\lceil T_{\text{start}} - (T_{\text{start}} - T_{\text{end}}) \times \frac{t_{\text{current}}}{t_{\text{limit}}} \right\rceil, \quad (13)$$

where $Q_{\text{start}}$ and $T_{\text{start}}$ are the parameter values at the beginning of the search, and $Q_{\text{end}}$ and $T_{\text{end}}$ are the values at the ending of the search. In addition, $t_{\text{current}}$ is the current elapsed time in a given moment of the search, and $t_{\text{limit}}$ is the maximum run time.

Depending on the $Q_{\text{values}}$ list, the search process can be conducted with large values of the destruction parameter in the initial iterations of the search process. After preliminary computational experiments, it could be observed that the use of a local search iteratively resulted in worse results. In this sense, the local search procedure is applied only before the while loop. A gradual reduction of the number of jobs removed in the current permutation and the temperature to accept worse solutions contribute to diversification in the first iterations and intensification in the last ones. Another factor to be highlighted is that, by using a range of values for the parameters of the algorithm, there is no need for statistical parameter tuning since the values are evaluated during the search process.

Initially, a permutation is generated using the EDD dispatch rule, which is improved using the NEH and 2-opt local search with the best improvement strategy. During a given run time, the following operators are repeated iteratively. First, the parameters $Q$ and $T$ are determined using Equations (12) and (13), respectively. Thereupon, the destruction and construction operators are applied. Finally, the generated solution is accepted based on the SA criterion, as expressed by Equation (11). Algorithm 2 illustrates the pseudocode of the introduced VIGD metaheuristic.

**Data:** $p$, $d$, $Q_{\text{start}}$, $Q_{\text{end}}$, $T_{\text{start}}$, $T_{\text{end}}$
**Result:** A schedule $\pi$
Generate an initial solution
Apply NEH algorithm
Apply local search
**while** *current_time* $\leq$ *time_limit* **do**
    Determine the parameters $Q$ and $T$.
    Apply destruction operator
    Apply construction operator
    Acceptance criterion
**end**

**Algorithm 2:** VIGD algorithm.

In the VIGD algorithm, the initial solution is improved with a 2-opt local search with the best improvement strategy. Thus, the entire neighbourhood is explored, and the best solution is returned. Taking this improved initial solution into account, the main loop of the VIGD is executed. Since different values for the destruction of the permutation are explored, the computational cost of this strategy is higher than the standard iterated greedy algorithm. Therefore, a local search procedure in the main loop of the VIGD algorithm is not considered.

## 5. Computational experiments

### 5.1. Experimental design

The test instances proposed by Ruiz and Allahverdi (2009) were evaluated. In such problems, the following parameters are considered: the number of machines ($m$), the number of jobs ($n$), the tardiness factor ($TF$), and the due date range ($RDD$). The total number of instance classes is given by 4 ($m$) × 5 ($n$) × 2 ($TF$) × 2 ($RDD$) = 80. Also, 10 test instances were randomly generated for each class, totalling 800 test instances. Since in the testbed proposed by Ruiz and Allahverdi (2009) there are no weights, integer weights were generated from a uniform distribution $U[1,10]$, as suggested by Silva, Valente, and Schaller (2022). Table 1 summarizes the characteristics of the evaluated test instances.

The Relative Deviation Index (RDI) is adopted as a performance measure, since it is a standard indicator for scheduling problems with due-date objectives (Fernandez-Viagas and Framinan 2015; Karabulut 2016). Let $H$ be a set of solution procedures, the $RDI$ returned by the solution procedure $s \in H$ when applied to instance $t$ is calculated as in Equation (14):

$$RDI_{st} = \begin{cases} 0, & \text{if } \min_{h \in H} T_{ht} = \max_{h \in H} T_{ht}, \\ \dfrac{T_{st} - \min_{h \in H} T_{ht}}{\max_{h \in H} T_{ht} - \min_{h \in H} T_{ht}} \times 100, & \text{otherwise,} \end{cases} \tag{14}$$

where $T_{st}$ is the weighted quadratic tardiness value obtained by method $s$ in instance $t$. In the present case, $\min_{h \in H} T_{ht}$ is the best solution found among the methods under comparison. To summarize

**Table 1.** Parameters of evaluated test instances.

| Parameter | Levels |
|---|---|
| Number of machines | $m \in \{5, 10, 15, 20\}$ |
| Number of jobs | $n \in \{20, 40, 60, 80, 100\}$ |
| Processing time distribution | $U[1,100]$ |
| Weight distribution | $U[1,10]$ |
| Tardiness factor | $TF \in \{0.0, 0.6\}$ |
| Due date ranges | $RDD \in \{0.2, 0.6\}$ |

the computational results, the Average RDI (ARDI) for a given method is calculated by grouping the RDI obtained across a given set of instances.

Furthermore, the Success Rate (SR) is considered as another performance indicator (Moccellin *et al.* 2018; Pitombeira-Neto and Athayde Prata 2020). SR is calculated as the number of times that a given method finds the best solution (with or without a draw) divided by the number of test instances in a given instance set, as expressed by Equation (15):

$$\mathrm{SR} = \frac{n_{\mathrm{BEST}}}{n_{\mathrm{INST}}} \times 100 \tag{15}$$

where $n_{\mathrm{BEST}}$ is the number of instances in which a given method achieved the best solution and $n_{\mathrm{INST}}$ is the number of instances in the given instance set.

For the MILP model, the commercial solver Gurobi (https://www.gurobi.com/) version 9.0.3 with the JuMP library (https://www.juliaopt.org/JuMP.jl/stable/) (Lubin and Dunning 2015) is used. Initially, the MILP model was run for all the evaluated test instances. It could be observed that the solutions provided by the MILP model taking the specified time limit into account were of low quality. Additionally, the MILP model failed to find a feasible integer solution in several test instances, so the MILP model was removed from the computational results.

Since the problem under study has been addressed before in the previous literature, the proposed metaheuristic method is compared with the best algorithms reported for the no-wait flowshop considering total tardiness-related objectives. All the metaheuristic algorithms are evaluated considering the objective function (1) and the problem constraints (2)–(10).

Three variants of the IG algorithm are considered. The first one (IG1) is the metaheuristic proposed by Ruiz and Allahverdi (2009) for the no-wait flowshop minimizing the makespan and maximum lateness. This algorithm was adapted for the problem under study considering the total tardiness minimization as the objective function to be minimized. The remainder IG algorithms (IG2 and IG3) are two parametrizations of the IG algorithm developed by Silva, Valente, and Schaller (2022) for a flowshop to minimize the weighted quadratic tardiness. In the present experiments, the Variable Greedy (VG) algorithm (Framinan and Leisten 2008) proposed by Silva, Valente, and Schaller (2022) for a flowshop is also considered in order to minimize the weighted quadratic tardiness. With respect to the RDIG, all the values suggested by Ruiz and Allahverdi (2007) were used to generate the $Q_{\mathrm{values}}$ list. Concerning the VIGD, the maximal and minimal values of $Q$ and $T$ were the same as those evaluated by Ruiz and Allahverdi (2007). The selected algorithms and the parameters adopted are summarized in Table 2. It can be highlighted that the parameters of the algorithms IG1, IG2, IG3 and VG are fixed. Concerning the RDIG, the parameter $T$ is fixed, and the parameter $Q$ is randomly selected from the $Q_{\mathrm{values}}$ list in each iteration. Finally, the parameters of the VIGD are determined in each iteration, as described in Equations (12) and (13).

All the methods were implemented using Julia version 1.6 (https://julialang.org/) with Visual Studio Code IDE (https://code.visualstudio.com/). The computational experience was performed on a PC with a 4.20 GHz Intel® Core™ i7-7700K CPU and 8 GB of memory, with the Ubuntu 20.04 LTS operating system. Since the methods under comparison are stochastic metaheuristics, each algorithm was run 10 times, and the average values were reported.

For all the evaluated methods, the following assumptions are considered for a fair comparison.

**Table 2.** Parameters used in the evaluated algorithms.

| Method | Parameters | Based on |
|---|---|---|
| IG1 | $Q = 4, T = 0.4$ | Ruiz and Allahverdi (2009) |
| IG2 | $Q = 7, T = 0.7, ls_{prob} = 1$ | Silva, Valente, and Schaller (2022) |
| IG3 | $Q = 4, T = 1.0, ls_{prob} = 0.1$ | Silva, Valente, and Schaller (2022) |
| VG | $size\_prop = 0.7, ls_{prob} = 0.7$ | Silva, Valente, and Schaller (2022) |
| RDIG | $T = 0.4, Q_{values} = \{2345678\}$ | the present authors' proposal |
| VIGD | $Q_{start} = 8, Q_{end} = 2, T_{start} = 0.5, T_{end} = 0.0$ | the present authors' proposal |

**Table 3.** Average and standard deviation of RDI for *n* and *m*.

| | | IG1 | | IG2 | | IG3 | | VG | | RDIG | | VIGD | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *m* | *n* | Avg. | St.Dev. | Avg. | St.Dev. | Avg. | St.Dev. | Avg. | St.Dev. | Avg. | St.Dev. | Avg. | St.Dev. |
| 5 | 20 | 86.2 | 26.1 | 69.9 | 31.7 | 21.5 | 31.2 | 33.3 | 27.4 | 73.7 | 27.6 | 7.1 | 18.7 |
| | 40 | 92.7 | 17.5 | 84.6 | 16.4 | 8.4 | 8.3 | 49.8 | 17.0 | 85.2 | 17.4 | 0.5 | 2.0 |
| | 60 | 88.7 | 24.1 | 79.0 | 26.6 | 11.2 | 8.7 | 54.7 | 19.1 | 81.6 | 27.6 | 0.3 | 1.1 |
| | 80 | 93.2 | 11.7 | 90.4 | 11.7 | 9.5 | 9.7 | 72.9 | 12.1 | 91.5 | 10.5 | 1.1 | 2.9 |
| | 100 | 89.2 | 20.4 | 84.8 | 22.8 | 5.2 | 6.3 | 76.8 | 20.1 | 85.4 | 22.5 | 4.1 | 16.0 |
| 10 | 20 | 87.5 | 22.4 | 66.8 | 24.7 | 7.0 | 10.5 | 32.6 | 23.3 | 75.5 | 22.3 | 2.8 | 5.6 |
| | 40 | 98.1 | 4.4 | 84.0 | 12.1 | 10.3 | 8.5 | 56.9 | 13.7 | 92.1 | 7.2 | 0.2 | 0.9 |
| | 60 | 95.3 | 10.1 | 84.6 | 14.6 | 14.2 | 8.2 | 63.6 | 12.1 | 94.2 | 10.8 | 0.0 | 0.0 |
| | 80 | 97.4 | 5.5 | 88.3 | 10.9 | 15.3 | 7.8 | 79.8 | 11.7 | 96.2 | 4.6 | 0.2 | 0.7 |
| | 100 | 96.6 | 5.0 | 90.9 | 8.9 | 18.4 | 5.8 | 86.4 | 10.4 | 95.3 | 5.5 | 0.0 | 0.0 |
| 15 | 20 | 91.7 | 20.3 | 64.1 | 26.5 | 10.7 | 17.7 | 41.5 | 26.0 | 79.1 | 19.1 | 1.8 | 4.2 |
| | 40 | 96.4 | 7.5 | 83.9 | 14.6 | 11.7 | 8.4 | 56.7 | 12.4 | 92.8 | 6.5 | 0.1 | 0.8 |
| | 60 | 96.7 | 5.5 | 91.7 | 10.2 | 12.8 | 6.5 | 76.7 | 9.6 | 93.0 | 7.9 | 0.2 | 0.8 |
| | 80 | 98.0 | 3.5 | 89.7 | 9.0 | 19.6 | 9.5 | 85.6 | 9.5 | 95.9 | 3.8 | 0.1 | 0.7 |
| | 100 | 96.5 | 5.6 | 92.6 | 9.5 | 23.0 | 9.1 | 90.8 | 10.6 | 93.8 | 8.1 | 0.0 | 0.0 |
| 20 | 20 | 93.0 | 13.1 | 64.3 | 29.7 | 11.2 | 18.1 | 35.2 | 22.3 | 80.8 | 19.6 | 4.0 | 13.3 |
| | 40 | 95.5 | 8.2 | 85.2 | 14.2 | 11.7 | 8.6 | 59.5 | 14.3 | 90.4 | 10.5 | 0.7 | 3.0 |
| | 60 | 98.3 | 3.2 | 90.0 | 9.5 | 16.5 | 8.5 | 78.7 | 12.0 | 93.0 | 7.7 | 0.0 | 0.0 |
| | 80 | 95.4 | 6.7 | 92.0 | 11.0 | 23.1 | 10.4 | 87.4 | 11.1 | 92.6 | 8.8 | 0.1 | 0.7 |
| | 100 | 95.4 | 5.7 | 94.2 | 7.2 | 32.0 | 8.4 | 92.9 | 6.5 | 96.1 | 5.3 | 0.0 | 0.0 |
| Min. | | 86.2 | 3.2 | 64.1 | 7.2 | 5.2 | 5.8 | 32.6 | 6.5 | 73.7 | 3.8 | 0.0 | 0.0 |
| Max. | | 98.3 | 26.1 | 94.2 | 31.7 | 32.0 | 31.2 | 92.9 | 27.4 | 96.2 | 27.6 | 7.1 | 18.7 |
| Avg. | | 94.1 | 11.3 | 83.6 | 16.1 | 14.7 | 10.5 | 65.6 | 15.1 | 88.9 | 12.7 | 1.2 | 3.6 |

- All algorithms are initialized with the dispatch rule. The jobs are scheduled by non-decreasing order of their due dates. With this sequence, the weighted quadratic tardiness is calculated.
- The 2-opt with best improvement policy is used as the local search procedure, as described in Section 4.2.
- The time limit $t_{limit} = n \times m/2 \times 60$ milliseconds is considered.

Aiming to measure the dispersion of the RDI values, the sample standard deviation $\sigma$ is calculated as in Equation (16), where $RDI_t$ is the RDI found for the test instance $t$, and $n_{INST}$ is the number of evaluated test instances:

$$\sigma = \frac{\sqrt{\Sigma(RDI_t - ARDI)^2}}{n_{INST} - 1}. \tag{16}$$

## 5.2. Results and discussion

Table 3 presents the results of average and standard deviation RDI values considering the number of machines and jobs. Based on the achieved results, the following comments can be summarized.

**Table 4.** Average and standard deviation of RDI for $T$ and $R$.

| | | IG1 | | IG2 | | IG3 | | VG | | RDIG | | VIGD | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | $R$ | Avg. | St.Dev. | Avg. | St.Dev. | Avg. | St.Dev. | Avg. | St.Dev. | Avg. | St.Dev. | Avg. | St.Dev. |
| 0.0 | 0.2 | 93.3 | 11.2 | 83.3 | 17.7 | 13.4 | 11.9 | 66.5 | 23.0 | 89.8 | 12.3 | 0.9 | 3.5 |
| | 0.6 | 89.0 | 20.4 | 80.8 | 23.0 | 12.6 | 11.0 | 63.1 | 25.5 | 84.8 | 21.0 | 1.3 | 7.6 |
| 0.6 | 0.2 | 97.3 | 8.3 | 85.2 | 19.2 | 16.3 | 14.5 | 65.6 | 26.4 | 91.8 | 13.4 | 1.1 | 6.1 |
| | 0.6 | 96.7 | 10.8 | 84.9 | 19.3 | 16.5 | 15.5 | 67.1 | 25.5 | 89.3 | 15.9 | 1.4 | 8.6 |
| Min. | | 89.0 | 8.3 | 80.8 | 17.7 | 12.6 | 11.0 | 63.1 | 23.0 | 84.8 | 12.3 | 0.9 | 3.5 |
| Max. | | 97.3 | 20.4 | 85.2 | 23.0 | 16.5 | 15.5 | 67.1 | 26.4 | 91.8 | 21.0 | 1.4 | 8.6 |
| Avg. | | 94.1 | 12.7 | 83.6 | 19.8 | 14.7 | 13.2 | 65.6 | 25.1 | 88.9 | 15.7 | 1.2 | 6.5 |

- IG1 returned the worst results in the most evaluated test instances. In average terms, IG1 is not competitive in comparison with all the other methods.
- IG2 returned better results than the IG1 algorithm. Nevertheless, this solution procedure is outperformed by the remainder of the algorithms under comparison.
- IG3 presented better objective function values than the IG1, IG2, VG and RDIG algorithms. On the other hand, this solution approach has not returned better results than the present authors' proposed VIGD algorithm.
- The VG algorithm presented better weighted squared tardiness than the IG1, IG2 and RDIG algorithms. However, this solution procedure is not better than IG3 and VIGD solution approaches.
- RDIG is not competitive for the evaluated test instances, only presenting better solutions than the IG1 algorithm. RDIG is almost equal to IG1. They only differ in the mechanism of selection of the jobs to be removed from the current permutation. It can be observed that the random selection for the destruction parameter of RDIG presented better results than the fixed strategy of IG1.
- The proposed VIGD presented the best results among all the other methods under comparison. It can be observed that VIGD returned ARDI values near zero on several sets of test problems. Furthermore, the VIGD algorithm presented the best solution for all instances with 10 machines and 100 jobs, 15 machines and 100 jobs, 20 machines and 60 jobs, and 20 machines and 100 jobs.

Table 4 describes the averages and standard deviations of RDI considering the tardiness factor $T$ and the range of due dates $R$. In view of such results, the following observations can be emphasized: all the methods under comparison demonstrated a greater difficulty in solving test instances with values of $T$ equal to zero than instances with $T$ equals to 0.6. Concerning the $R$ values, no clear trend is observable.

Based on the results described in Tables 3 and 4, it can be observed that the ARDI values of IG1, IG2, IG3, VG and RDIG are much higher than those of VIGD. This is because the VIGD algorithm explores distinct values of the parameters of destruction and temperature throughout the search. In the first iterations of the search process, the VIGD algorithm selects high values for Q and T, which implies greater diversification. In the last iterations, smaller values for Q and T are selected, resulting in greater intensification. This strategy based on a VND algorithm has proved to be effective to improve the results of a standard IG.

Table 5 describes the average SR for $n$ values. On the basis of the achieved results, the following points can be highlighted.

- IG1, IG2, and RDIG presented average SR values smaller than 1.0%. Thus, the number of test instances in which such solution approaches are competitive is reduced.
- VG presented a slightly better average success rate than the previous solution approaches; however, this value still is much lower than the average values returned by IG3 and VIGD algorithms.

**Table 5.** Average SR for *n* values.

| *n* | IG1 | IG2 | IG3 | VG | RDIG | VIGD |
|---|---|---|---|---|---|---|
| 20 | 0.6 | 2.5 | 36.3 | 5.0 | 0.0 | 65.6 |
| 40 | 0.0 | 0.0 | 8.8 | 0.6 | 0.0 | 91.9 |
| 60 | 1.3 | 1.3 | 5.6 | 1.9 | 0.6 | 95.0 |
| 80 | 0.0 | 0.0 | 7.5 | 0.0 | 0.0 | 92.5 |
| 100 | 0.6 | 0.6 | 10.6 | 0.6 | 0.6 | 89.4 |
| Min. | 0.0 | 0.0 | 5.6 | 0.0 | 0.0 | 65.6 |
| Max. | 1.3 | 2.5 | 36.3 | 5.0 | 0.6 | 95.0 |
| Avg. | 0.5 | 0.9 | 13.8 | 1.6 | 0.3 | 86.9 |

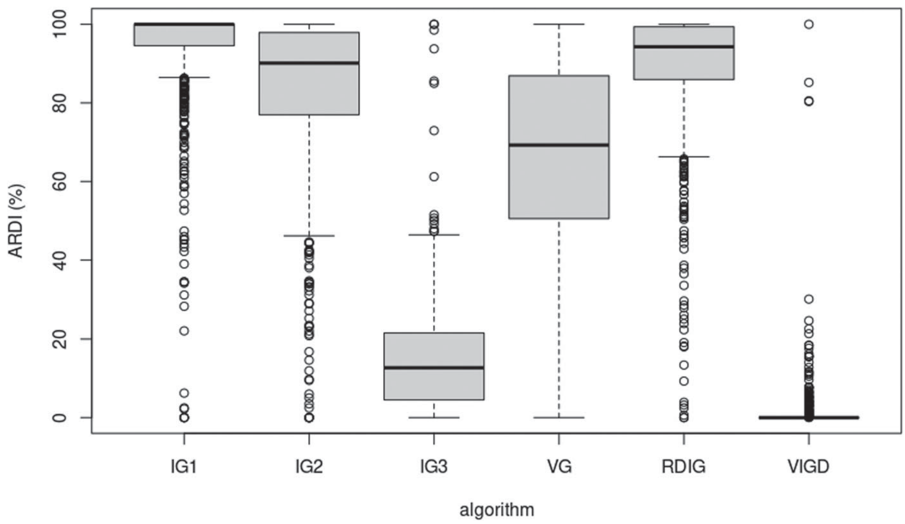


**Figure 2.** Boxplots for average RDI values.

- IG3 presented better average success rate values than the previous algorithms. The best behaviour of this algorithm occurred in the test problems with 20 jobs, where the SR was approximately 36%. In the test problems with 40, 60, 80, and 100 jobs, the average success rates are closer to 10%.
- The proposed VIGD returned the better average success rate values for all the sets of instances. For the test problems with 40, 60, 80, and 100 jobs, the obtained success rates are closer to 90%.

Aiming to evaluate if the difference among the solution procedures under comparison is statistically significant, ANOVA and Tukey tests were performed. Figure 2 illustrates the boxplot for ARDI values, and Figure 3 illustrates the Tukey confidence intervals for ARDI values.

Figure 2 describes a pairwise comparison between all the methods under evaluation. From this figure, the following results can be emphasized:

- The IG1 algorithm returned worse results than all the other methods under comparison.
- The IG2 algorithm returned better results only than the IG1 algorithm.
- The difference between the results of RDIG and VG is not statistically significant.
- The IG3 algorithm is outperformed only by the VIGD.
- The VIGD algorithm outperformed all the other algorithms under study.

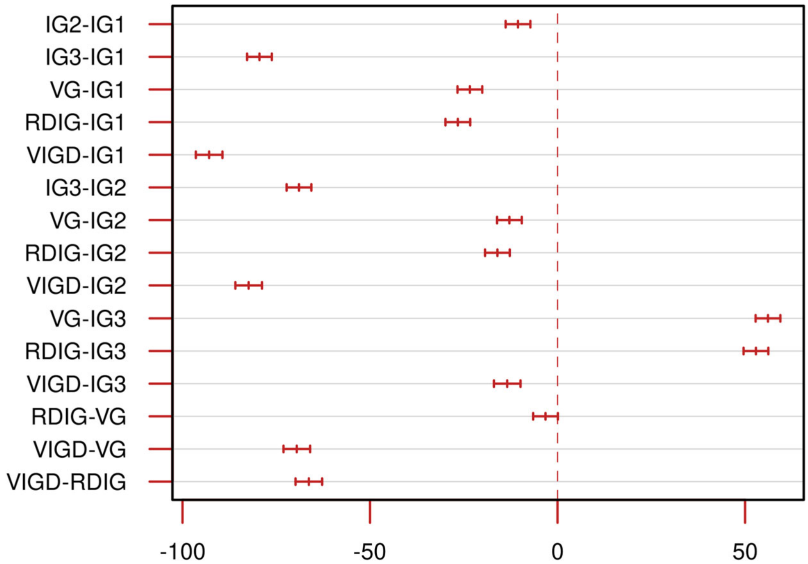


**Figure 3.** Tukey confidence intervals for ARDI values.

The ANOVA test returned a statistics $F = 1956$. Since this value is much higher than the critical value of 4.39, the difference among the solution procedures is statistically significant. With respect to the presented statistical analysis, it can be observed that the proposed VIGD outperformed all the other solution approaches under comparison with statistically significant differences.

## 6. Final remarks and perspectives

In this article, a no-wait permutation flowshop with weighted quadratic tardiness minimization is introduced. A mixed-integer linear programming formulation with positional decision variables and a quadratic objective function is presented. First, an iterated greedy algorithm that randomly selects the destruction parameter througout the search process is developed. Furthermore, a novel iterated greedy metaheuristic that dynamically variates the parameters of the number of jobs to be removed from the current sequence and the temperature is proposed.

Computational experiments were carried out to evaluate the performance of the proposed algorithms. The relative deviation index and the success rate are used as performance measures. Computational times were not applied for comparison purposes since all the methods under comparison were run with the same time limit. In most evaluated test instances, the IG1 metaheuristic provided the worst solutions. On the other hand, the proposed VIGD presented the best solution in most of the evaluated instances.

As extensions of this work, the evaluation of the proposed VIGD in other combinatorial optimization problems (Sakuraba and Yagiura 2010; Sakuraba *et al.* 2015) can be recommended. Another research avenue is the consideration of weighted quadratic tardiness in other permutation flowshop variants. Furthermore, future studies could also consider other strategies for evaluating the parameters of the iterated greedy algorithms.

## Disclosure statement

The authors declare that they have no conflict of interest.

## Funding

This study was financed in part by the Coordination for the Improvement of Higher Education Personnel (CAPES) and the National Council for Scientific and Technological Development (CNPq) [grants 404232/2016-7, 303594/2018-7, 306075/2017-2, 430137/2018-4, 312585/2021-7, 407151/2021-4].

## Availability of data and materials

The data sets are available at http://soa.iti.es/problem-instances. The results of all computational tests are available at https://www.researchgate.net/publication/360040702_Results_-_A_novel_iterated_greedy_algorithm_for_the_no-wait_permutation_flowshop_scheduling_-to_minimize_weighted_quadratic_tardiness?showFulltext=1&linkId=625eb87d4173a21-a0d1ec944.

## ORCID

*Bruno de Athayde Prata* http://orcid.org/0000-0002-3920-089X*Marcelo Seido Nagano* http://orcid.org/0000-0002-0239-1725

## References

Aldowaisan, Tariq, and Ali Allahverdi. 2012. "Minimizing Total Tardiness in No-Wait Flowshops." *Foundations of Computing and Decision Sciences* 37 (3): 149–162.
Aldowaisan, Tariq, and Ali Allahverdi. 2015. "No-Wait Flowshops to Minimize Total Tardiness with Setup Times." *Intelligent Control and Automation* 6 (1): 38–44. doi:10.4236/ica.2015.61005
Allahverdi, Ali. 2016. "A Survey of Scheduling Problems with No-Wait in Process." *European Journal of Operational Research* 255 (3): 665–686.
Allahverdi, Ali, Harun Aydilek, and Asiye Aydilek. 2020. "No-Wait Flowshop Scheduling Problem with Separate Setup Times to Minimize Total Tardiness Subject to Makespan." *Applied Mathematics and Computation* 365: Article ID 124688. doi:10.1016/j.amc.2019.124688
Costa, Maria Raquel C., Jorge Valente, and Jeffrey E. Schaller. 2020. "Efficient Procedures for the Weighted Squared Tardiness Permutation Flowshop Scheduling Problem." *Flexible Services and Manufacturing Journal* 32 (3): 487–522.
Della Croce, Federico, Andrea Grosso, and Fabio Salassa. 2021. "Minimizing Total Completion Time in the Two-Machine No-Idle No-Wait Flow Shop Problem." *Journal of Heuristics* 27 (1): 159–173.
Ding, Jian-Ya, Shiji Song, Jatinder N. D. Gupta, Rui Zhang, Raymond Chiong, and Cheng Wu. 2015. "An Improved Iterated Greedy Algorithm with a Tabu-Based Reconstruction Strategy for the No-Wait Flowshop Scheduling Problem." *Applied Soft Computing* 30: 604–613.
Ding, Jianya, Shiji Song, Rui Zhang, Jatinder N. D. Gupta, and Cheng Wu. 2015. "Accelerated Methods for Total Tardiness Minimisation in No-Wait Flowshops." *International Journal of Production Research* 53 (4): 1002–1018.
Fernandez-Viagas, Victor, and Jose M. Framinan. 2015. "NEH-Based Heuristics for the Permutation Flowshop Scheduling Problem to Minimise Total Tardiness." *Computers & Operations Research* 60: 27–36. doi:10.1016/j.cor.2015.02.002.
Framinan, Jose Manuel, and Rainer Leisten. 2008. "Total Tardiness Minimization in Permutation Flow Shops: A Simple Approach Based on a Variable Greedy Algorithm." *International Journal of Production Research* 46 (22): 6479–6498.
Gonçalves, Tomás C., Jorge M. S. Valente, and Jeffrey E. Schaller. 2016. "Metaheuristics for the Single Machine Weighted Quadratic Tardiness Scheduling Problem." *Computers & Operations Research* 70: 115–126.
Hansen, Pierre, and Nenad Mladenović. 2001. "Variable Neighbourhood Search: Principles and Applications." *European Journal of Operational Research* 130 (3): 449–467.
Hoitomt, Debra J., Peter B. Luh, Eric Max, and Krishna R. Pattipati. 1990. "Scheduling Jobs with Simple Precedence Constraints on Parallel Machines." *IEEE Control Systems Magazine* 10 (2): 34–40.
Jackson, James R. 1955. *Scheduling a Production Line to Minimize Maximum Tardiness*. Technical Report. Management Science Research Project, Numerical Analysis Research Document. National Technical Information Service Issue Number 198024. California University Los Angeles. https://ntrl.ntis.gov/NTRL/dashboard/searchResults/titleDetail/AD152722.xhtml.
Karabulut, Korhan. 2016. "A Hybrid Iterated Greedy Algorithm for Total Tardiness Minimization in Permutation Flowshops." *Computers & Industrial Engineering* 98: 300–307.
Lin, Shih-Wei, Chung-Cheng Lu, and Kuo-Ching Ying. 2018. "Minimizing the Sum of Makespan and Total Weighted Tardiness in a No-Wait Flowshop." *IEEE Access* 6: 78666–78677.
Lubin, Miles, and Iain Dunning. 2015. "Computing in Operations Research Using Julia." *INFORMS Journal on Computing* 27 (2): 238–248.
Luh, Peter B., and Debra J. Hoitomt. 1993. "Scheduling of Manufacturing Systems Using the Lagrangian Relaxation Technique." *IEEE Transactions on Automatic Control* 38 (7): 1066–1079.

Miyata, Hugo Hissashi, Marcelo Seido Nagano, and Jatinder N. D. Gupta. 2019. "Integrating Preventive Maintenance Activities to the No-Wait Flow Shop Scheduling Problem with Dependent-Sequence Setup Times and Makespan Minimization." *Computers & Industrial Engineering* 135: 79–104.

Mladenović, Nenad, and Pierre Hansen. 1997. "Variable Neighbourhood Search." *Computers & Operations Research* 24 (11): 1097–1100.

Moccellin, Joao Vitor, Marcelo Seido Nagano, Anselmo Ramalho Pitombeira Neto, and Bruno de Athayde Prata. 2018. "Heuristic Algorithms for Scheduling Hybrid Flow Shops with Machine Blocking and Setup Times." *Journal of the Brazilian Society of Mechanical Sciences and Engineering* 40 (2): 1–11.

Muller, Laurent Flindt, Simon Spoorendonk, and David Pisinger. 2012. "A Hybrid Adaptive Large Neighbourhood Search Heuristic for Lot-Sizing with Setup Times." *European Journal of Operational Research* 218 (3): 614–623.

Nagano, Marcelo Seido, Fernando Siqueira de Almeida, and Hugo Hissashi Miyata. 2021. "An Iterated Greedy Algorithm for the No-Wait Flowshop Scheduling Problem to Minimize Makespan Subject to Total Completion Time." *Engineering Optimization* 53 (8): 1431–1449.

Nawaz, Muhammad, E. Emory Enscore Jr, and Inyong Ham. 1983. "A Heuristic Algorithm for the $m$-Machine, $n$-Job Flow-Shop Sequencing Problem." *Omega* 11 (1): 91–95.

Pitombeira-Neto, Anselmo Ramalho, and Bruno de Athayde Prata. 2020. "A Matheuristic Algorithm for the One-Dimensional Cutting Stock and Scheduling Problem with Heterogeneous Orders." *Top* 28: 178–192. doi:10.1007/s11750-019-00531-3.

Ruiz, Rubén, and Ali Allahverdi. 2007. "No-Wait Flowshop with Separate Setup Times to Minimize Maximum Lateness." *The International Journal of Advanced Manufacturing Technology* 35 (5): 551–565.

Ruiz, Rubén, and Ali Allahverdi. 2009. "New Heuristics for No-Wait Flow Shops with a Linear Combination of Makespan and Maximum Lateness." *International Journal of Production Research* 47 (20): 5717–5738. doi:10.1080/00207540802070942

Ruiz, Rubén, and Thomas Stützle. 2007. "A Simple and Effective Iterated Greedy Algorithm for the Permutation Flowshop Scheduling Problem." *European Journal of Operational Research* 177 (3): 2033–2049.

Sakuraba, Celso S., Débora P. Ronconi, Ernesto G. Birgin, and Mutsunori Yagiura. 2015. "Metaheuristics for Large-Scale Instances of the Linear Ordering Problem." *Expert Systems with Applications* 42 (9): 4432–4442.

Sakuraba, Celso S., and Mutsunori Yagiura. 2010. "Efficient Local Search Algorithms for the Linear Ordering Problem." *International Transactions in Operational Research* 17 (6): 711–737.

Schaller, Jeffrey, and Jorge M. S. Valente. 2012. "Minimizing the Weighted Sum of Squared Tardiness on a Single Machine." *Computers & Operations Research* 39 (5): 919–928.

Schaller, Jeffrey, and Jorge M. S. Valente. 2018. "Efficient Heuristics for Minimizing Weighted Sum of Squared Tardiness on Identical Parallel Machines." *Computers & Industrial Engineering* 119: 146–156.

Silva, Andreia F., Jorge M. S. Valente, and Jeffrey E. Schaller. 2022. "Metaheuristics for the Permutation Flowshop Problem with a Weighted Quadratic Tardiness Objective." *Computers & Operations Research* 140: Article ID 105691. doi:10.1016/j.cor.2021.105691

Stafford Jr, Edward F., and Fan T. Tseng. 2002. "Two Models for a Family of Flowshop Sequencing Problems." *European Journal of Operational Research* 142 (2): 282–293.

Stafford Jr, E. F., Fan T. Tseng, and Jatinder N. D. Gupta. 2005. "Comparative Evaluation of MILP Flowshop Models." *Journal of the Operational Research Society* 56 (1): 88–101. doi:10.1057/palgrave.jors.2601805

Sun, Xiaoqing, James S. Noble, and Cerry M. Klein. 1999. "Single-Machine Scheduling with Sequence Dependent Setup to Minimize Total Weighted Squared Tardiness." *IIE Transactions* 31 (2): 113–124.

Thomalla, Christoph S. 2001. "Job Shop Scheduling with Alternative Process Plans." *International Journal of Production Economics* 74 (1-3): 125–134.

Valente, Jorge M. S., and Jeffrey E. Schaller. 2012. "Dispatching Heuristics for the Single Machine Weighted Quadratic Tardiness Scheduling Problem." *Computers & Operations Research* 39 (9): 2223–2231.

Ye, Honghan, Wei Li, Amin Abedini, and Barrie Nault. 2017. "An Effective and Efficient Heuristic for No-Wait Flow Shop Production to Minimize Total Completion Time." *Computers & Industrial Engineering* 108: 57–69.

Zhao, Fuqing, Huan Liu, Yi Zhang, Weimin Ma, and Chuck Zhang. 2018. "A Discrete Water Wave Optimization Algorithm for No-Wait Flow Shop Scheduling Problem." *Expert Systems with Applications* 91: 347–363.

Zhao, Fuqing, Shuo Qin, Guoqiang Yang, Weimin Ma, Chuck Zhang, and Houbin Song. 2019. "A Factorial Based Particle Swarm Optimization with a Population Adaptation Mechanism for the No-Wait Flow Shop Scheduling Problem with the Makespan Objective." *Expert Systems with Applications* 126: 41–53.