International Journal of Industrial Engineering Computations 12 (2021) 415–426

Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

Minimization of total tardiness in no-wait flowshop production systems with preventive maintenance

Tuane Tonani Yamada[a], Marcelo Seido Nagano[a*] and Hugo Hissashi Miyata[a]

[a]University of São Paulo, São Carlos School of Engineering, Production Engineering Department, Av. Trabalhador São-carlense, 400, 13566-590, São Carlos, São Paulo, Brazil

C H R O N I C L E

Article history:
Received February 18 2021
Received in Revised Format
April 31 2021
Accepted May 3 2021
Available online
May, 4 2021

Keywords:
No-wait flowshop
Preventive maintenance
Total tardiness
Heuristic methods

A B S T R A C T

Efficient business organizations must balance quality, cost, and time constraints in competitive environments. Reflecting the complexity of this task, we consider manufacturing systems including several stages of production chains requiring time measurement. When production scheduling is not prioritized in such enterprises, several negative effects may occur. A corporation may suffer financial penalties as well as negative brand exposure, and thus may find its credibility challenged. Therefore, in this study, we propose constructive methods to minimize a total tardiness criterion, considering preventative maintenance constraints to reflect the reality of industrial practice, focusing on a no-wait flowshop environment in which jobs are successively processed without operational interruptions. In addition to proposing constructive methods to solve the no-wait flowshop production scheduling problem, a metaheuristic is presented as an approach to improve results obtained by constructive methods. Computational experiments were designed and performed to compare several production scheduling algorithms. Among various constructive heuristics considered, an algorithm called HENLL using an insertion logic showed the best performance. The proposed metaheuristic is based on the iterated greedy (IG) search method, and the results obtained demonstrated significant improvement compared to the heuristics alone. It is expected that this study may be used by production planning and control (PPC) professionals to apply the proposed method to schedule production more efficiently. We show that the proposed method successfully presented a better solution in relation to total tardiness, considering the above mentioned environment.

1. Introduction

Production scheduling is among the most important activities in the operations of manufacturing systems, because it is responsible for scheduling jobs on machines and specifying the times each job must be performed (Ruiz et al., 2007). MacCarthy and Liu (1993) proposed a general scheduling problem as follows. n jobs $\{J_1,J_2,...,J_n\}$ must be processed on m available machines $\{M_1,M_2,...,M_m\}$. A subset of these machines is required to process each job. The default flow (machine order) may or may not be fixed for all jobs. The processing of a job J_j job on a machine M_i is called an operation, denoted by O_{ij}. A p_{ij} time is associated with the processing of each operation O_{ij}. A delivery date variable d_i denotes the time when J_j must be finalized. In this context, scheduling consists of job assignment over time on machines. The question regarding scheduling is to find appropriate methods to optimize a performance measure (Maccarthy & Liu, 1993). According to Pinedo (2008), a scheduling problem can be specified according to an $\alpha|\beta_i|\gamma$ notation, where α represents the machine environment, β_i is defined as additional constraints of the problem, and γ represents the objective function adopted. The problem addressed in this study is defined as a no-wait flowshop scheduling problem with the goal of minimizing the total tardiness under a preventive maintenance restriction specified as $F_m|no\text{-}wait, MP(m)|\sum_i T_i$.

* Corresponding author Tel.: 55 16 3373-9428; Fax: 55 16 3373-9425
E-mail: drnagano@usp.br drnagano@sc.usp.br (M. S. Nagano)

2021 Growing Science Ltd.
doi: 10.5267/j.ijiec.2021.5.002

The no-wait flowshop problem has been studied since the 1970s (Reddi & Ramamorthy, 1972; Wismer, 1972). In general, it consists of processing n jobs on m machines in the same order continuously with no interruptions. Hall and Sriskandarajah (1996) pointed out that an instance of a no-wait flowshop problem may occur because of some characteristics of the material being processed, such as temperature and viscosity, which require an operation to follow immediately after the previous one. This may be observed in steelmaking.

Macchiaroli et al. (1999) pointed out that the no-wait flowshop problem is important for manufacturing because of the technological constraints that manufacturing processes face. Moreover, it is understood to reduce work in process (WIP). Recently, there has been an increase in the number of studies devoted to the problem of inefficiencies, which may be expressed as $(F_m|no\text{-}wait|\sum_j T_j)$. Supply chain system delays are understood to be directly related to payment of fines and loss of customers (Ding et al, 2015). Ruiz and Allahverdi (2006) studied the no-wait flowshop problem with setup times independent of processing times and considered a maximum tardiness minimization problem which may be expressed as $\left(F_m\middle|no\text{-}wait,s_{ij}\middle|L_{max}\right)$. The authors proposed constructive heuristics and four new genetic algorithms (GAs). Aldowaisan and Allahverdi (2012) proposed dispatch rules and heuristic methods for the $F_m|no\text{-}wait|\sum_i T_i$ problem, formulating modified due date (MDD), single machine processing time (SMPT) and earliest due date with processing time (EDDP) rules, generating an initial solution for their proposed simulated annealing (SA) and a GA. To improve the SA and GA methods, the researchers applied the PAAH heuristic. Liu, Song, and Wu (2013) proposed some heuristics for the $F_m|no\text{-}wait|\sum_i T_i$ problem. The authors tested dispatch rules based on problem data, such as processing time and delivery date, to generate an initial solution for the NEH heuristic second phase. Arabameri and Salmasi (2013) proposed a mixed-integer linear programming model for the no-wait flowshop problem with sequence-dependent setup times and an objective function minimizing weighed delay and an advance sum $\left(F_m\middle|no\text{-}wait,s_{ij}\middle|\sum_j\left(W_jE_j + W_jT_j\right)\right)$. The authors proposed the use of the Tabu search (TS) and particle swarm optimization (PSO) methods, and found that the latter presented better results. To investigate the no-wait flowshop problem to minimize total tardiness with independent setup times, Aldowaisan and Allahverdi (2015) proposed heuristic methods for the $F_m\middle|no\text{-}wait,s_{ij}\middle|\sum_j T_j$ problem. The authors proposed dispatch rules for the problem and further developed improvements for SA and GA heuristics. The MDD dispatch rule was used to improve the initial SA heuristic solution. GAs, MDD, EDDP, and total tardiness based on averages across all machines (TOTA) were used to compose the initial population. An insertion algorithm was also proposed to develop improved versions of SA and GAs (Aldowaisan & Allahverdi, 2015). Ding et al. (2015) studied new properties and proposed heuristic methods for the $F_m|no\text{-}wait|\sum_j T_j$ problem, developing an accelerated NEH algorithm and a greedy heuristic (Ding et al., 2015). Most of the problems studied assume that machines are always available. However, in practice, deterministic and random factors such as preventive maintenance affect the availability of production machines (Pinedo, 2008).

Preventive maintenance and production scheduling are closely related activities, even though they are treated separately in the literature. Maintenance generally includes techniques or operations that allow machines to be kept in service. Preventive maintenance decreases the probability that production systems may fail unexpectedly. Maintenance activities are typically scheduled to enable machines to perform to a certain level of reliability. Both job scheduling and maintenance tasks require significant labor time. This implies a competition between multiple time priorities, which must be balanced for an industrial plant to function well and perform reliably (Ruiz et al., 2007).

Some authors who studied no-wait flowshop problems considering preventive maintenance in the context of alternate objective functions have proposed solutions with constructive heuristics (Miyata et al., 2019a, 2019b). Parameterization of heuristics with preventive maintenance was found to be essential for better results (Miyata et al., 2019a; Perez-Gonzalez et al., 2020).
The remainder of this study is organized as follows. In Section 2, we present the basic concepts of the problem addressed. In Section 3, several proposed constructive heuristics are discussed. In Section 4, several proposed metaheuristics are presented, and Section 5 provides the results of computational experiments conducted to verify the efficacy of the proposed approach. Final conclusions are presented in Section 6.

2. Basic concepts

2.1 Preventive maintenance

Maintenance operations may be divided into two primary groups. Corrective maintenance (CM) includes any maintenance performed when equipment has already failed. Preventive maintenance (PM) occurs on systems still in operation, as an action taken before the equipment fails (Ruiz et al., 2007).

To improve the reliability of a system, PM policies need to be established and adopted, as machines are exposed to corrosion, fatigue, wear, and other variables (Cheng et al., 2017). Predictive maintenance is a helpful tool and occurs more often when PM policies incorporate information about the current state of machines, in which case the prediction of failure becomes more accurate (Cai et al., 2013).

According to Ruiz et al. (2007), PM policies should be based on statistical theories of reliability. The Weibull distribution is widely used to represent machine failure time, as failure rates are not constant in this distribution. Ruiz et al. (2007) mentioned three policies, aiming to maximize machine availability or maintain minimum levels of production reliability.

- Policy 1 (fixed-time PM model): Programmers adjust the schedule to include pauses between production cycles for maintenance so as not to interrupt one or a set of production orders.
- Policy 2 (PM model with machine availability maximization): The purpose of this policy is to maximize system availability. Thus, the ideal interval between PM (T_{PM2}) can be planned according to the equation below, where the parameter *tr* is the repair time, *tp* is the time spent on PM, β = form, and θ = scale.

$$T_{PM2} = \theta \times \left[\frac{t_p}{t_r(\beta - 1)}\right]^{1/\beta}, \tag{1}$$

As an example, suppose the machine failure time follows the Weibull distribution $X \approx [\theta,\beta]$, with $\theta = 1200$, $\beta = 2$, $t_r = 4$ hours and $t_p = 1$ hour. Applying Equation (1), we find that the optimal T_{MP} interval is 600 hours.

- Policy 3 (PM model maintaining a minimum reliability for a given production period): This policy aims to perform systematic PM, that is, at regular intervals. The equation used for this policy (T_{PM3}) is as follows.

$$T_{PM3} = \left[-\theta^{\beta} * \frac{\ln R0(t)}{t}\right]^{1/(\beta-1)}, \tag{2}$$

where $Ro(t)$ is a reliability function.

Suppose the machine failure time follows the Weibull model $X \approx [\theta,\beta]$, where $\theta = 1200$ and $\beta = 2$. The goal, then, is to determine the time interval T_{MP} such that the reliability after t = 670 h is $R(t)$ = 90%. Applying Equation (2), we find that T_{MP} = 226,45 hours, which means that for approximately every 227 h, a PM activity is performed.

2.2 No-wait flowshop problem with preventive maintenance and total tardiness minimization

The $F_m|no\text{-}wait, MP(m)|\sum_i T_i$ problem can be defined as follows. A set of n jobs $\{j=1,...,n\}$ must be processed on a set of m machines $\{i=1,...,m\}$, arranged in series. Jobs follow a standard processing flow across the machines. Each job j has an operation associated with a machine i, defined as $O_{j,i}$ with a processing time $p_{j,i}$. To satisfy the no-wait condition, jobs must be processed continuously. Each job has a delivery date d_j. The goal of the problem is to find a sequence of jobs such that total tardiness, as the sum of the tardiness times of all jobs, is minimized. All machines are subject to PM, as determined by the policies outlined in Section 2.1. The level of machine degradation is given by the difference between the PM interval and the operating time in which machines process jobs. When the degradation level of a machine j reaches a value insufficient to process the next job, PM is performed, and its level returns to a maximum. The level of degradation cannot be allowed to reach a value below zero, as in practice, this could pose a high risk of damage and, in theory, a machine breakdown.

To ensure that jobs are processed without interruption, the idle time between j and k jobs ($D_{j,k}$) on the first machine, that is, $D_{j,k}$, represents the distance between the start of the processing of a job j and the start of the processing of the subsequent job k (Bertolissi, 2000).

$$D_{jk} = \max_{1 \le i \le m}\left(\sum_{h=1}^{i} p_{h,j} - \sum_{h=1}^{i-1} p_{h,k}\right), \tag{3}$$

where p_{ij} is the processing time of a job j on a machine i.

This distance can be calculated for all possible job pairs in the sequence forming a matrix $D_{j,k}$ of $n \times n$ size. Based on distance information, Bertolissi (2000) proposed calculating the completion time of a job j in a sequence π ($C_{[j]}\pi$) according to the following expression.

$$C_{[J]}(\pi) = \sum_{l=2}^{i} D_{[l-1.l]}(\pi) + P_{[j]}(\pi), \tag{4}$$

where, $P_{[j]}(\pi) = \sum_{i=1}^{m} p_{[j],i}$.

Thus, the sequence total tardiness time can be calculated as follows (Liu et al., 2013).

$$\sum_j T_j(\pi) = \sum_{j=1}^{n} max\{C_{[j]}(\pi) - d_{[j]}(\pi), 0\}, \tag{5}$$

where d_j is the job delivery date, occupying the j position of the π sequence.

To include PM stops, it is necessary to calculate the distance between the finish date of a job z and the start date of a job j for each machine. This distance is called clearance ($Dclearance_{izj}$). Figure 1 graphically demonstrates these distances.

$$Dclearance_{izj} = DI_{ij} - DT_{iz}, \quad (6)$$

where DI denotes the start date, DT the end date, and j and z are jobs on a machine i where $j = z+1$.

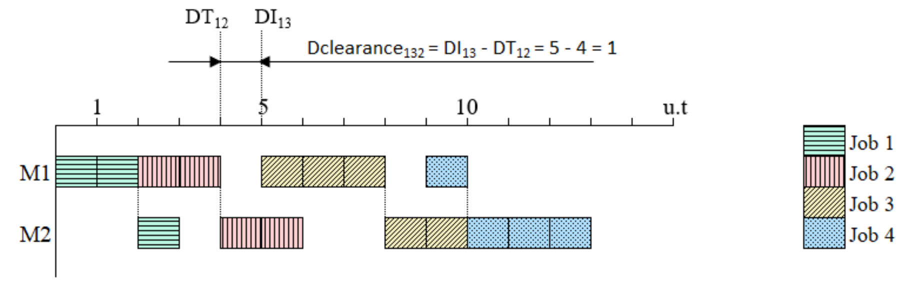

Fig. 1. Distance among jobs

With these calculated distances between all jobs on every machine, the time required to perform PM can be verified. If PM needs to be allocated within a timeframe occupied by a job, the number of time units the job must be shifted must be checked for PM to be performed without violating the no-wait criterion.

To include PM in the analyzed methods (comparative or proposed), the algorithms were first performed without it, and once the final sequence with the shortest total tardiness was generated, PM was added. In the next section, the proposed algorithms for solving the problem described are presented.

3. Proposed constructive heuristics

No methods simultaneously addressing total tardiness in a no-wait flowshop environment considering PM characteristics and constructive heuristics were found in the literature. Therefore, a method with some of the desired features other than PM, was selected as a heuristic for comparison with the proposed methods. Hence, adaptation had a low impact because it did not imply any change in the methodology proposed by the original authors. This constructive heuristic, developed by Liu et al. (2013), was used to compare the performance of the proposed heuristics. The authors named the method MNEH as it uses a modified NEH algorithm. MNEH can be described as follows.

- Sort the jobs according to an EDD rule to generate an initial sequence of jobs.
- Sort the first two jobs and select the job with the lowest total tardiness. In the case of a tie, select the smallest makespan.
- Run the NEH algorithm against the least total tardiness criterion. In the case of a tie, select the smallest makespan.

3.1. HIN proposed algorithm

The first proposed algorithm was inspired by the initial solution in the FPDNEH heuristic of Ronconi and Henriques (2009) and its method of generation. The first step of the FPDNEH algorithm is to calculate the Ij term for all jobs j, after which the lowest job Ij is selected for the first position. In the proposed heuristic, called HIN, all jobs are arranged in ascending order in relation to Ij, and this sequence is identified as the initial solution. The term Ij is a composition of delivery dates and processing times on the first machine, as it directly affects the total tardiness value, that is

$$I_j = d_j + p_{j1}, \quad (7)$$

where dj is the delivery date of a job j and pj_1 is the processing time of job j on the first machine.

The next step is to apply the NEH algorithm considering the lowest total tardiness generated in the candidate sequences. Fig. 2 shows pseudocode describing the HIN heuristic.

```
Given a set J = $\{j_1, j_2, j_3, \dots, j_n\}$ of $n$ jobs, where $S$ is the scheduled job set, and $R$ is the unscheduled job set, perform the following steps.

Step 1: $R \leftarrow J$ ; $S \leftarrow \Phi$.
Step 2: Arrange the jobs of $R$ in ascending order of $I_j$ ($I_j = d_j + p_{j1}$).
Step 3: Create a substring with the lowest $I_j$ job; call this substring $S$.
Step 4: Include the jobs that are not in the set S in the set R.
Step 5: Select the first job of the set $R$ set and test it in all possible positions in the set $S$ without
        changing the relative positions of jobs that already belong to the set S (NEH
        mechanism). Select the subsequence with the lowest total tardiness (T), where $T = \sum max\ \{Cj - dj. 0\}$. If there is a tie in total tardiness time, select the subsequence with
        the lowest makespan.
Step 6: Repeat step 5 until $S \leftarrow \Phi$.
Step 7: Calculate total tardiness (T) of the final sequence $S$.
```

Fig. 2. HIN algorithm

3.2. HMN proposed algorithm

Aldowaisam and Allahverdi (2012) addressed the no-wait flowshop problem to minimize total tardiness. First, the authors studied various dispatch rules (EDD, EDDP, MDD, SLACK, SRMWK, SMPT, and ASMPT), among which MDD excelled. Subsequently, they used simulated annealing (SA) metaheuristics and genetic algorithms (GA) to solve the problem.

As the main interest of the present work is the study of constructive heuristics, we propose a method referred to as HMN consisting of an operation of sequencing the jobs according to the MDD rule and applying the insertion phase of the NEH algorithm. Pseudocode describing the proposed heuristic is given in Fig. 3.

```
Given a set J = $\{j_1, j_2, j_3, \dots, j_n\}$ of $n$ jobs, where $S$ is the scheduled job set and $R$ is the unscheduled job set, perform the following steps.

Step 1: $R \leftarrow J$ ; $S \leftarrow \Phi$.
Step 2: Arrange the jobs of $R$ in ascending order of MDD, where MDD = max$\{Cj. dj\}$.
Step 3: Create two subsequences of two jobs, the first subsequence comprising the first two jobs
        of $R$ ($S' \leftarrow R_1.R_2$ ), and the second subsequence comprising the first two jobs of $R$ in
        reverse order ($S'' \leftarrow R_2.R_1$ ).
Step 4: Calculate total tardiness (T) for the sequences $S'$ e $S''$, and select the sequence with the
        lowest T, where $T = \sum max\ \{Cj - dj. 0\}$, and call it $S$.
Step 5: Include the jobs that are not in the set S in the set R.
Step 6: Select the first job of the set $R$ and test it in all possible positions in the set $S$ without
        changing the relative positions of jobs that already belong to S (NEH mechanism).
        Select the subsequence with the lowest total tardiness. In the case of a tie in total
        tardiness time, select the subsequence with the lowest makespan.
Step 7: Repeat step 6 until $R \leftarrow \Phi$.
Step 8: Calculate total tardiness (T) of the $S$ final sequence.
```

Fig. 3. HMN algorithm

3.3. HENN proposed algorithm

The article published by Naderi and Arshadi (2013) and Ding et al. (2015) inspired the next proposed method.

Naderi and Arshadi (2013) studied a no-wait flowshop with the aim of minimizing total tardiness. The techniques used to solve the problem included variable neighborhood search (VNS) and simulated annealing algorithm. The VNS was subdivided into two stages, referred to as NSS1 and NSS2.

- NSS1: One job is removed at a time randomly (without repetition), reinserted in all positions, and the system is then tested to determine whether total tardiness decreased. If the sequence generated by NSS1 is accepted, another job is then randomly selected to perform the same process again.
- NSS2: Two jobs are randomly selected and randomly inserted into another position. This procedure is tested n (parameter) times until two other jobs are randomly selected to perform the same process. The best solution is accepted even if it is inferior to the current solution. Thus, the optimum sequence can be found.

```
Given a set J = $\{j_1, j_2, j_3, \dots, j_n\}$ of $n$ jobs, where $S$ is the scheduled job set and $R$ is the unscheduled job set, perform the following steps.

Step 1: $R \leftarrow J$ ; $S \leftarrow \Phi$.
Step 2: Arrange the jobs $R$ in ascending order of delivery date (EDD rule).
Step 3: Create two subsequences of two jobs, the first subsequence comprising the first two jobs
        of $R$ ($S' \leftarrow R_1.R_2$ ), and the second subsequence comprising the first two jobs of $R$, in
        reverse order ($S'' \leftarrow R_2.R_1$ ).
Step 4: Calculate total tardiness (T) for the $S'$ and $S''$ sequences, and select the sequence with the
        lowest $T$, where $T = \sum max\ \{Cj - dj. 0\}$, and call it $S$.
Step 5: Include the jobs that are not in the set $S$ in the set $R$.
Step 6: Select the first job of the set $R$ and test it in all possible positions in the set $S$ without
        changing the relative positions of the jobs that already belong to the set $S$ (NEH
        mechanism). Select the subsequence with the lowest total tardiness. If there is a tie in
        total tardiness time, select the subsequence with the lowest makespan.
Step 7: Repeat step 6 until $R \leftarrow \Phi$.
Step 8: Consider $j = 1$.
Step 9: Remove the job $j$ and apply the NEH procedure. Select the sequence with the lowest total
        tardiness and call it $S$.
Step 10: If $j \neq n$, consider $j = j + 1$ and go to step 9, otherwise go to step 11.
Step 11: Calculate total tardiness (T) of the final sequence $S$.
```

Fig. 4. HENN algorithm

Ding et al. (2015) addressed the problem of total tardiness minimization in the context of a no-wait flowshop, and the technique used encompassed two accelerated heuristics. First, an initial solution was obtained using the EDD dispatch rule. Next, the first two jobs are evaluated, the one with the lowest total tardiness is selected, and the sequence is completed with the NEH job insertion phase. The second heuristic destroys and builds a previously obtained sequence. Thus, jobs are randomly removed and added to the partial sequence through the NEH procedure. Therefore, the proposed algorithm referred to as HENN was inspired by works by Naderi and Arshadi (2013) and Ding et al. (2015). First, an initial solution is generated using the EDD dispatch rule, the first two jobs are evaluated, and the NEH algorithm is applied. Finally, NSS1 is applied without the random factors job by job (first to job number 1, then to job number 2, and so forth until all jobs have been verified) and then removed to test if total tardiness decreased on the other positions; if so, the sequence is accepted. Jobs are removed according to their position in the original dataset, and not according to the position of the last *generated* sequence. For clarity, Fig. 4 shows pseudocode.

3.4. HENLL proposed algorithm

The next proposed heuristic was based on the metaheuristic developed by Ribas et al. (2013), who studied a blocking flowshop scheduling problem with total tardiness criteria. The procedure begins with an NEH algorithm, where the initial solution is a sequence generated by the EDD or FPD rules.

According to the authors, both dispatch rules performed similarly to generate an initial solution. Therefore, the generated sequence was improved by local search algorithms, first by LS1 for the purpose of swapping, and second by LS2 for the purpose of insertion. LS1 randomly selects a job and swaps positions with a job ahead of it in the sequence, whereas LS2 randomly selects a job and inserts it in all positions, regardless of whether it is located before or after the original position of the selected jobs. For both, a new candidate sequence is only accepted if the total tardiness is lower than the total tardiness of the current sequence. The last step presented by the authors was a disturbance mechanism that allowed the escape of optimum placements.

As a proposal for a constructive heuristic based on Ribas (2013), we examine the HENLL algorithm, which is composed of an NEH phase from the initial solution generated by the EDD rule, and the evaluation of the first two jobs. LS1 and LS2 phases were applied before insertion as described by Ribas (2013), but without the random factors. Moreover, insertion was applied instead of exchange in the LS1 step, that is, the jobs selected for insertion followed an ascending order, first testing job number 1, then job number 2, and then repeating the process until all jobs are tested. Pseudocode describing this algorithm is shown in Fig. 5.

```
Given a set J = {$j_1, j_2, j_3, \dots, j_n$} of $n$ jobs, where $S$ is the scheduled job set, $R$ is the unscheduled job set, perform the following steps.

Step 1: $R \leftarrow J$ ; $S \leftarrow \Phi$.
Step 2: Arrange the jobs $R$ in ascending order of delivery date (EDD rule).
Step 3: Create two subsequences of two jobs, the first subsequence comprising the first two jobs
        of $R$ ($S' \leftarrow R_1.R_2$ ), and the second the first two jobs of $R$ in reverse order ($S'' \leftarrow R_2.R_1$ ).
Step 4: Calculate total tardiness (T) for the sequences $S'$ and $S''$ sequences, and select the
        sequence with the lowest $T$, where $T = \sum max\ \{Cj - dj.0\}$, and call it $S$.
Step 5: Include the jobs that are not in the set S in the set R.
Step 6: Select the first job from the set $R$ and test it in all possible positions in the set $S$ without
        changing the relative positions of jobs that already belong to the set S (NEH
        mechanism). Select the subsequence with the lowest total tardiness. If there is a tie in
        total tardiness time, select the subsequence with the lowest makespan.
Step 7: Repeat step 6 until $R \leftarrow \Phi$.
Step 8: Consider $j = 1$.
Step 9: Remove the job $j$ and apply the NEH procedure, except that the test position should be
        greater than the original position of $j$. Select the sequence with lowest total tardiness and
        call it $S$.
Step 10: If $j \neq n$, consider whether $j = j + 1$ and if so, go to step 9; otherwise go to step 11.
Step 11: Consider $j = 1$.
Step 12: Remove the job $j$ and apply the NEH procedure. Select the sequence with the lowest
         total tardiness and call it $S$.
Step 13: If $j \neq n$, consider whether $j = j + 1$ and if so, go to step 12; otherwise go to step 14.
Step 14: Calculate total tardiness (T) of the final sequence $S$.
```

Fig. 5. HENLL algorithm

4. Proposed metaheuristics

To improve the studied constructive heuristics, we propose the elaboration of a metaheuristic by adapting the constructive heuristic with the shortest computational time to transforming it into a metaheuristic, and thus compare it with the best constructive heuristic. The proposed metaheuristic is based on the method developed by Ruiz and Stutzle (2007), in which the initial solution consists of a descending order of processing times. Next, the first two jobs are selected such that the two possible partial sequences can be evaluated. The other jobs are entered according to their initial sequence through the NEH procedure. The generated sequence feeds the iterated greedy (IG) algorithm, which consists of a destruction phase in which some jobs are deleted, and a construction phase in which the deleted jobs are reinserted one by one in the order they were deleted by the NEH algorithm. The number of jobs eliminated is a parameter, and the algorithm only ends when an acceptance criterion is met. The proposed metaheuristic was inspired by the iterated greedy (IG) procedure, and it comprises the solution generated by the MNEH algorithm plus the result of the IG algorithm. The method is called MHNIG, and three parameter settings were tested, as given below.

- MHNIG[1] includes a parameterizable number of jobs deleted in the IG destruction phase. This number of jobs (*Rem*) is a function of the number of machines and jobs in the problem, as given by the following equation.

$$Rem = roundup\left(\frac{j}{m/2}\right), \tag{8}$$

where *Rem* is the number of jobs deleted, *j* is the number of jobs, *and m* is the number of machines. In addition to the parameterizable number of jobs deleted, setting [1] includes stopping criteria with *factor* = 1, where factor is a component of the loop equation

$$Loop = roundup\left(\frac{j}{Rem}\right) * factor. \tag{9}$$

That is, if *Rem* = 2 and the number of jobs in the problem is 10, we have *loop* = 5. Thus, the IG procedure is performed 5 times, because *factor* = 1. For this same example, if the *factor* were equal to 10, the IG procedure would be performed 50 times.

- MHNIG[2] includes a fixed number of jobs deleted during the destruction phase, namely 3 jobs, and *factor* = 1.
- MHNIG[3] includes a fixed number of jobs deleted during the destruction phase, namely 3 jobs, and *factor* = 10.

The purpose of these settings is to test the influence of parameterization (depending on the number of jobs and machines) on the number of jobs deleted and the impact of increase by *factor*. Fig. 6 shows pseudocode for the MHNIG method.

```
Given a set $J = \{j_1, j_2, j_3, \dots, j_n\}$ of $n$ jobs, where $S$ is the scheduled job set and $R$ is the unscheduled job set, perform the following steps.
Step 1: $R \leftarrow J$ ; $S \leftarrow \Phi$.
Step 2: Arrange the $R$ jobs in ascending order of delivery date (EDD rule).
Step 3: Create two subsequences of two jobs, the first comprising the first two jobs of $R$ ($S' \leftarrow R_1.R_2$ ),
        and the second subsequence comprising the first two jobs of $R$ in reverse order ($S'' \leftarrow R_2.R_1$ ).
Step 4: Calculate total tardiness ($T$) for the $S'$ and $S''$ sequences, and select the sequence with the smallest
        $T$, where $T = \sum max\ \{Cj - dj.0\}$, and call it $S$.
Step 5: Include the jobs that are not in the set $S$ in the set $R$.
Step 6: Select the first job from the set $R$ and test it in all possible positions in the set $S$ without changing
        the relative positions of the jobs that already belong to the set S (NEH mechanism). Select the
        subsequence with the lowest total tardiness. If there is a tie in total tardiness time, select the
        subsequence with the lowest makespan.
Step 7: Repeat step 6 until $R \leftarrow \Phi$.
Step 8: Calculate the number of jobs to be deleted ($Rem$) and the number of loops ($Loop$).
Step 9: Consider $LoopQty = 1$.
Step 10: Delete $Rem$ jobs at random from the $S$ sequence and place them in the sequence $R$. Apply NEH
        procedure to $R \leftarrow \Phi$. Select the sequence with lowest total tardiness and call it $S$.
Step 11: If $LoopQty \neq Loop$, consider $LoopQty = LoopQty + 1$ and go to step 10, otherwise go to step
        12.
Step 12: Calculate the total tardiness ($T$) of the final sequence $S$.
```

Fig. 6. MHNIG algorithm

5. Computational experiments

All methods were performed on a system with an Intel(R) Core(TM) i7-75000U (2.70 GHz and 2.90 GHz) CPU and 8GB of RAM. For the scenarios, the available bases recognized in the literature were used. Ronconi and Henriques (2009) provided instances where processing times were evenly distributed between 1 and 99, and delivery dates were evenly distributed between *P(1-T-R/2)* and *P(1-T+R/2)*, where *T* and *R* are the job tardiness factor and delivery date dispersion interval, respectively, and *P* is related to the characteristics of an unlimited buffer flowshop. As the problem of interest is a PM flowshop, it was necessary to adapt the delivery dates and *P* factor to accommodate these restrictions. Thus, *P* is defined as t_{vj} = processing time on machine *v* of job *j*, where *m* denotes a machine, r_{vx} = PM time *x* number on machine *v*. Thus, the term r_{vx} includes the condition of PM to generate the delivery dates, as it considers the maximum number of maintenance events on each machine. The number of maintenance instances per machine was defined according to policies 1, 2, and 3.

$$P = \max\left\{ \max_{1\le v\le m} \begin{bmatrix} \sum_{j=1}^{n} tvj + \min\limits_{j} \sum_{q=1}^{v-1} tqj + \\ \min\limits_{j} \sum_{q=v+1}^{m} tqj + \sum_{x=1}^{w} rvx \end{bmatrix} .\ \max_{j} \sum_{v=1}^{m} tvj \right\}. \tag{10}$$

120 instances were generated with 4 delivery date scenarios each, that is, the *T* and *R* parameters were varied in the following ways.

- Scenario 1 (c1): T=0.2 and R=0.6.
- Scenario 2 (c2): T=0.2 and R=0.8.
- Scenario 3 (c3): T=0.4 and R=0.6.
- Scenario 4 (c4): T=0.4 and R=0.8.

The higher the parameter T, the lower the tardiness values that must be respected, and the higher the parameter *R*, the greater the amplitude range of tardiness values that must be respected.

Each heuristic was tested on 480 instances (120 issues with 4 delivery date scenarios each), ranging from 20 jobs and 5 machines to 500 jobs and 200 machines. In addition, three PM scenarios were applied, totaling 1,440 instances for each heuristic tested. The following logic was used to generate the PM values.

- Policy 1 (fixed-time interval PM model): At every 100 units of time processed, machines are stopped for PM, and maintenance durations were generated from a uniform distribution ranging from 1 to 25 units of time.
- Policy 2 (PM model with maximization of machine availability): The T_{PM2} equation was applied with parameters $\theta = 1500$, $\beta = 2$, $tp = 1$, and $tr = 8$, and the result obtained was a stop at every 530 time units processed per machine (T_{PM2}). A uniform distribution ranging from 1 to 25 units of time was used to generate the PM durations.

- Policy 3 (PM model maintaining minimum reliability for a given production period): The T_{PM3} equation was applied with the parameters $\theta = 1500$, $\beta = 2$, $t = 800$, and $Ro(t) = 0\ 85$, and the result obtained was a stop at every 457 time units processed per machine (T_{PM3}). A uniform distribution of 1 to 25 units of time was used to generate the PM durations.

The relative deviation measure described as follows was used to compare the results of the proposed heuristics.

$$Relative\ Deviation_{\ Best} = \frac{Rproposed\ \text{-}\ Rbest''}{Rbest} *100\%, \tag{11}$$

where *Rbest* is the result of the best algorithm, and *Rproposed* is the result of the proposed algorithm.

When *Rbest* = 0 and *Rproposed* > 0, we obtain a result of + ∞. In this situation, the example was not considered when calculating the average of the results.

5.1. Analysis of heuristic results

The results obtained by PM policies are given below in Tables 1, 2, and 3. The HENLL algorithm showed a superiority to the other algorithms because it uses insertion techniques which evaluate whether an insertion was beneficial. It is possible to verify that the use of more than one technique in the composition of the algorithm tended to improve the result.

Table 1
Relative deviation of heuristics with preventive maintenance policy 1

Problem	MNEH	HIN	HMN	HENN	HENLL
j20 | m5	24.3%	32.0%	24.3%	3.3%	1.3%
j20 | m10	8.3%	8.1%	8.1%	1.0%	0.3%
j20 | m20	5.6%	5.1%	4.9%	1.7%	0.8%
j50 | m5	36.9%	42.8%	36.9%	4.7%	2.6%
j50 | m10	14.0%	14.9%	14.0%	2.4%	0.6%
j50 | m20	7.4%	7.5%	7.1%	1.6%	0.3%
j100 | m5	44.0%	46.8%	44.0%	5.8%	1.5%
j100 | m10	17.7%	17.7%	17.7%	3.8%	0.2%
j100 | m20	8.6%	8.7%	8.6%	1.9%	0.3%
j200 | m10	17.4%	17.3%	17.4%	2.9%	0.2%
j200 | m20	8.6%	8.7%	8.6%	1.5%	0.3%
j500 | m20	8.2%	8.3%	8.2%	1.6%	0.0%
Average	16.8%	18.2%	16.7%	2.7%	0.7%

Table 2
Relative deviation of heuristics with preventive maintenance policy 2

Problem	MNEH	HIN	HMN	HENN	HENLL
j20 | m5	23.7%	27.5%	24.0%	3.8%	0.8%
j20 | m10	12.3%	14.5%	12.7%	2.4%	1.3%
j20 | m20	4.3%	4.3%	4.9%	0.9%	0.8%
j50 | m5	34.9%	37.0%	34.9%	4.9%	1.9%
j50 | m10	16.5%	14.9%	16.5%	3.3%	0.4%
j50 | m20	7.7%	7.3%	7.8%	1.6%	0.3%
j100 | m5	39.3%	42.9%	39.3%	6.2%	2.7%
j100 | m10	16.8%	15.6%	16.8%	2.4%	0.6%
j100 | m20	9.1%	9.4%	9.1%	1.6%	0.3%
j200 | m10	16.3%	16.4%	16.3%	2.8%	0.2%
j200 | m20	8.0%	7.9%	8.0%	1.6%	0.3%
j500 | m20	8.2%	8.2%	8.2%	1.8%	0.1%
Average	16.4%	17.2%	16.5%	2.8%	0.8%

To obtain a better result, we found that it was essential to use an initial solution appropriate to the type of problem to be solved. As the experiments show, the initial EDD solution facilitates the resolution of problems involving total tardiness, because it considers the delivery date variable. Regarding computational times, algorithms that have more complex insertion heuristics require more processing time. Thus, the HENN and HENLL algorithms had the longest processing times. The other algorithms using the technique of generating an initial solution and then applying NEH similarly required significantly less computational time. The following figures show the computational time of the algorithms used per policy tested (Figs. 7, 8, and 9).

Table 3
Relative deviation of heuristics with preventive maintenance policy 3

Problem	MNEH	HIN	HMN	HENN	HENLL
j20 | m5	30.7%	30.0%	30.7%	4.6%	3.2%
j20 | m10	11.7%	11.7%	11.7%	2.0%	1.0%
j20 | m20	4.4%	5.4%	4.4%	1.1%	0.5%
j50 | m5	35.3%	35.6%	35.3%	4.4%	2.8%
j50 | m10	16.0%	17.1%	16.0%	3.6%	0.5%
j50 | m20	6.8%	6.8%	6.9%	1.4%	0.8%
j100 | m5	40.7%	41.3%	40.7%	5.0%	1.2%
j100 | m10	19.0%	17.8%	19.0%	2.6%	0.8%
j100 | m20	8.8%	8.7%	8.8%	1.4%	0.6%
j200 | m10	15.7%	17.2%	15.7%	2.6%	0.5%
j200 | m20	8.0%	7.8%	8.0%	1.7%	0.2%
j500 | m20	8.7%	8.6%	8.7%	2.1%	0.1%
Average	17.1%	17.3%	17.1%	2.7%	1.0%

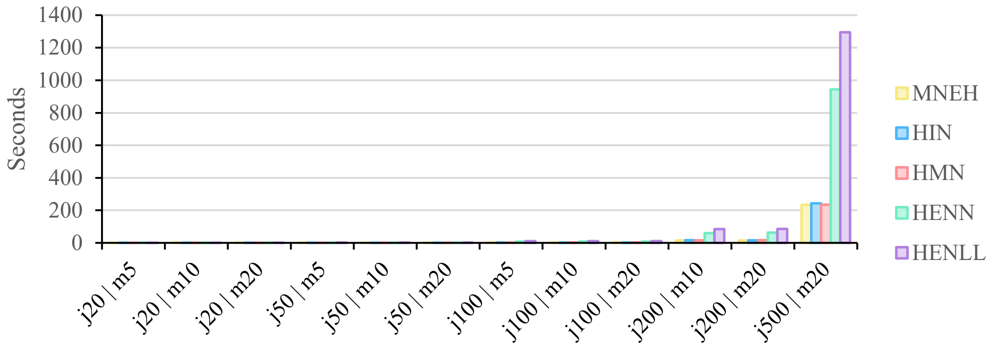

Fig. 7. Graphs of construction heuristics' execution times with preventive maintenance policy 1

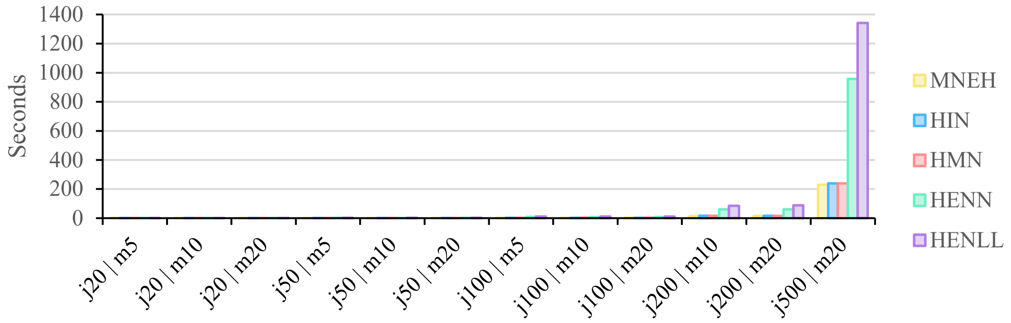

Fig. 8. Graphs of construction heuristics' execution times with preventive maintenance policy 2

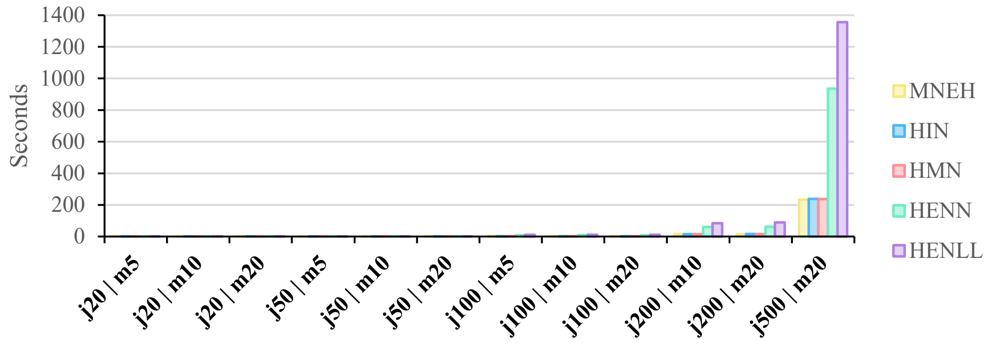

Fig. 9. Graphs of construction heuristics' execution times with preventive maintenance policy 3

5.2. Analysis of metaheuristic results compared to the best heuristic

Results comparing the performance of the proposed metaheuristics in contrast with the best heuristic previously identified are shown below in Table 4. Regarding PM policies, number 1 was applied. The results show that modifying the calculation for the number of jobs to be parameterizable did not yield good results, whereas a small and fixed number of jobs improved the results. A smaller number of jobs removed prevents the destruction of the solution previously obtained, in other words, by removing few tasks, the benefits obtained in the current solution are preserved. However, effective improvement was only achieved when more loops were employed, which was the case with the MHNIG algorithm [3], as more candidate solutions were tested.

Table 4
Relative deviation of heuristics and metaheuristics with preventive maintenance policy 1

Problem	HENLL	MHNIG[1]	MHNIG[2]	MHNIG[3]
j20 | m5	4.5%	29.2%	17.4%	4.9%
j20 | m10	3.6%	7.8%	7.3%	2.6%
j20 | m20	2.2%	4.5%	4.3%	1.2%
j50 | m5	7.4%	44.0%	19.0%	2.8%
j50 | m10	5.6%	14.3%	10.9%	1.2%
j50 | m20	2.3%	6.3%	4.9%	1.3%
j100 | m5	2.9%	53.0%	21.7%	4.0%
j100 | m10	3.7%	15.5%	10.7%	0.5%
j100 | m20	2.9%	7.5%	6.3%	0.3%
j200 | m10	3.5%	15.6%	9.8%	0.1%
j200 | m20	2.3%	6.8%	5.8%	0.1%
j500 | m20	1.3%	7.2%	5.1%	0.1%
Average	3.5%	17.6%	10.3%	1.6%

Thus, it was possible to obtain better results than those presented by HENLL. However, the metaheuristics must be optimally parameterized, otherwise the results may be expected to be poor.

6. Conclusion

Solving no-wait flowshop problems considering PM with the aim of minimizing total tardiness has been a challenge in many industries such as steel and food production, among many others, which need to solve this problem to improve their operational efficiency. This study proposes heuristic solutions to this problem. Among the proposed heuristics, that entitled HENLL obtained the best result among the constructive heuristics; however, its required computational time was also the highest. Its performance was achieved because of the insertion mechanisms executed by the algorithm. Insertion allows new sequences to be obtained while also preserving the relative position of the jobs, except for the job to be inserted. Thus, the previously generated sequence is not completely destroyed. The condition is that the initial solution must be the best solution, as this is expected to affect the performance of the entire algorithm. The initial solution used (EDD) has proven to be a good solution for problems requiring total tardiness minimization, as it considers the delivery date; that is, jobs with the latest delivery dates are scheduled first.

In addition to the constructive heuristic, a metaheuristic was proposed, as there were no references to the problem studied. Thus, the best constructive heuristic was compared with the MHNIG metaheuristics. The results show that, depending on metaheuristic settings, this approach may demonstrate higher performance than that of HENLL. The increased computational time is acceptable given the benefit of reducing total tardiness. Removing fewer jobs positively affects the MHNIG performance. Therefore, it is assumed that the removal of many jobs significantly destroys the previous ordering and does not leads to minimization of total tardiness. The number of insertion loops to be executed was determined to be another direct positive factor.

Regarding PM policies, no specific behavior indicating any correlation between the results and PM policies was found. The manner in which PM was considered in the algorithm contributed to this result, as PM was scheduled after the final sequence was identified. If the PM constraint on job ordering was considered because the initial solution was reached, the correlation between maintenance policies and total tardiness results could be clearer.

This study discussed production scheduling problems in the context of a no-wait flowshop, where the goal is to minimize the total tardiness. It is expected that the best-performing algorithm can be used by PPC professionals to solve production scheduling problems, especially when it is desired to minimize total tardiness.

Finally, we suggest including a PM variable in the creation of a heuristic as well as the elaboration of metaheuristics in future research. We further suggest using the HENLL heuristic as the initial solution.

Acknowledgement

This work was supported by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) - Brazil, under grant numbers 306075/2017-2, 430137/2018-4, and 160347/2019-0.

References

Aldowaisan, T. A., & Allahverdi, A. (2012). No-wait flowshop scheduling problem to minimize the number of tardy jobs. *The International Journal of Advanced Manufacturing Technology*, *61*(1-4), 311-323.

Aldowaisan, T., & Allahverdi, A. (2015). No-wait flowshops to minimize total tardiness with setup times. *Intelligent Control and Automation, 6*(01), 38.

Arabameri, S., & Salmasi, N. (2013). Minimization of weighted earliness and tardiness for no-wait sequence-dependent setup times flowshop scheduling problem. *Computers & Industrial Engineering, 64*(4), 902-916..

Bertolissi, E. (2000). Heuristic algorithm for scheduling in the no-wait flow-shop. *Journal of Materials Processing Technology, 107*(1-3), 459-465.

Cai, Y., Hasenbein, J. J., Kutanoglu, E., & Liao, M. (2013). Single-machine multiple-recipe predictive maintenance. *Probability in the Engineering and Informational Sciences, 27*(2), 209-235.

Cheng, G. Q., Zhou, B. H., & Li, L. (2017). Joint optimization of lot sizing and condition-based maintenance for multi-component production systems. *Computers & Industrial Engineering*, *110*, 538–549.

Ding, J., Song, S., Zhang, R., Gupta, J. N., & Wu, C. (2015). Accelerated methods for total tardiness minimisation in no-wait flowshops. *International Journal of Production Research, 53*(4), 1002-1018.

Hall, N. G., & Sriskandarajah, C. (1996). A survey of machine scheduling problems with blocking and no-wait in process. *Operations Research, 44*(3), 510-525.

Liu, G., Song, S., & Wu, C. (2013). Some heuristics for no-wait flowshops with total tardiness criterion. *Computers & Operations Research, 40*(2), 521-525..

Maccarthy, B. L., & Liu, J. (1993). Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling. *The International Journal of Production Research, 31*(1), 59-79.

Macchiaroli, R., Mole, S., & Riemma, S. (1999). Modelling and optimization of industrial manufacturing processes subject to no-wait constraints. *International Journal of Production Research, 37*(11), 2585-2607.

Miyata, H. H., Nagano, M. S., & Gupta, J. N. (2019). Incorporating preventive maintenance into the m-machine no-wait flow-shop scheduling problem with total flow-time minimization: a computational study. *Engineering Optimization, 51*(4), 680-698.

Miyata, H. H., Nagano, M. S., & Gupta, J. N. (2019b). 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.

Naderi, B., & Arshadi, A. (2013). Scheduling a variant of flowshop problems to minimize total tardiness. *Technical Journal of Engineering and Applied Sciences*, *3*, 3142–3149.

Perez-Gonzalez, P., Fernandez-Viagas, V., & Framinan, J. M. (2020). Permutation flowshop scheduling with periodic maintenance and makespan objective. *Computers & Industrial Engineering*, *143*, 106369.

Pinedo, M. L. (2008). *Scheduling: Theory, algorithms, and systems*. New York: Springer.

Reddi, S. S., & Ramamoorthy, C. V. (1972). On the flow-shop sequencing problem with no wait in process. *Journal of the Operational Research Society*, *23*(3), 323-331.

Ribas, I., Companys, R., & Tort-Martorell, X. (2013). An efficient iterated local search algorithm for the total tardiness blocking flow shop problem. *International Journal of Production Research, 51*(17), 5238-5252.

Ronconi, D. P., & Henriques, L. R. (2009). Some heuristic algorithms for total tardiness minimization in a flowshop with blocking. *Omega, 37*(2), 272-281.

Ruiz, R., & Allahverdi, A. (2007). No-wait flowshop with separate setup times to minimize maximum lateness. *The International Journal of Advanced Manufacturing Technology, 35*(5), 551-565.

Ruiz, R., García-Díaz, J. C., & Maroto, C. (2007). Considering scheduling and preventive maintenance in the flowshop sequencing problem. *Computers & Operations Research, 34*(11), 3314-3330.

Ruiz, R., & Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research, 177*(3), 2033-2049.

Wismer, D. A. (1972). Solution of the flowshop-scheduling problem with no intermediate queues. *Operations research*, *20*(3), 689-697.