Computers and Operations Research 81 (2017) 203–215

Contents lists available at ScienceDirect

Computers and Operations Research

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

Algorithms for job scheduling problems with distinct time windows and general earliness/tardiness penalties

Bruno Ferreira Rosa[a], Marcone Jamilson Freitas Souza[b], Sérgio Ricardo de Souza[a,*], Moacir Felizardo de França Filho[a], Zacharie Ales[c], Philippe Yves Paul Michelon[c]

[a] Federal Center of Technological Education of Minas Gerais (CEFET-MG), +55-31-3319-6780, ZIP Code: 30510-000, Belo Horizonte, MG, Brazil
[b] Department of Computer Science, Federal University of Ouro Preto (UFOP), ZIP Code: 35400-000, Ouro Preto, MG, Brazil
[c] LIA EA 4128, University of Avignon and Pays de Vaucluse (UAPV), ZIP Code: 84911, Avignon, France

ARTICLE INFO

Article history:
Received 7 October 2015
Revised 5 September 2016
Accepted 27 December 2016
Available online 28 December 2016

Keywords:
Scheduling
Makespan
Idle time
Enumeration algorithm

ABSTRACT

This paper addresses the single machine scheduling problem with distinct time windows and sequence-dependent setup times. The objective is to minimize the total weighted earliness and tardiness. The problem involves determining the job execution sequence and the starting time for each job in the sequence. An implicit enumeration algorithm denoted IE and a general variable neighborhood search algorithm denoted GVNS are proposed to determine the job scheduling. IE is an exact algorithm, whereas GVNS is a heuristic algorithm. In order to define the starting times, an $O(n^2)$ idle time insertion algorithm (ITIA) is proposed. IE and GVNS use the ITIA algorithm to determine the starting time for each job. However, the IE algorithm is only valid for instances with sequence-independent setup times, and takes advantage of theoretical results generated for this problem. Computational experiments show that the ITIA algorithm is more efficient than the only other equivalent algorithm found in the literature. The IE algorithm allows the optimal solutions of all instances with up to 15 jobs to be determined within a feasible computational time. For larger instances, GVNS produces better-quality solutions requiring less computational time compared with the other algorithm from the literature.

1. Introduction

The emergence of the *just in time* (JIT) management system highlighted the importance of carefully planning production activities. Reducing earliness and tardiness in job scheduling may result in significant cost reductions. According to Baker and Scudder [4], completing a job with tardiness, i.e., after the desired completion date, may result in contractual penalties, a loss of credibility for the company, and reduced sales. Similarly, completing a job before the desired date may result in extra financial costs, due to requirements for early capital availability, storage space or other resources for the maintenance and management of the inventory [17].

The single machine scheduling problem with distinct time windows and sequence-dependent setup times, as discussed in the present article, consists of sequencing and determining the time within which jobs must be performed in order to minimize the weighted sum of earliness and tardiness penalties in the execution of the jobs. This is an NP-hard problem [2], and is hereafter denoted by SMSPETP.

According to Wan and Yen [31], it is expected in many manufacturing industry situations that the jobs are finished within a certain time interval (time window), rather than at a single point in time (due date), because of uncertainties and tolerances. Such uncertainties and tolerances are related to individual job characteristics that influence the size of these time windows. Thus, only jobs completed before or after their respective time windows will be subject to penalties.

There exist a variety of applications of time window scheduling models in JIT manufacturing, semi-conductor manufacturing, chemical processing, PERT/CPM scheduling, and video on demand services, among others [16]. For instance, consider the production of perishable goods, as presented in Koulamas [20]. Assume that a chemical manufacturer combines a certain chemical "*A*", which deteriorates rapidly, with a second chemical "*B*", to produce a chemical "*C*". If "*A*" is produced before "*B*" is ready, then it will deteriorate. If "*A*" is produced later, then delays in the production of "*C*" may prove costly.

In industries where different product types are manufactured and the types of jobs processed on the machine frequently change,

* Corresponding author. .
E-mail addresses: brunorosa@div.cefetmg.br (B.F. Rosa), marcone@iceb.ufop.br (M.J.F. Souza), sergio@dppg.cefetmg.br (S.R. de Souza), franca@des.cefetmg.br (M.F. de França Filho), zacharie.ales@univ-avignon.fr (Z. Ales), philippe.michelon@univ-avignon.fr (P.Y.P. Michelon).

http://dx.doi.org/10.1016/j.cor.2016.12.024
0305-0548/

it is usually necessary to set up the machine between the execution of consecutive jobs [2]. The setup time includes the time spent changing tools, preparing the material, cleaning the machine, etc. Most studies regarding scheduling problems assume that setup times are independent of the execution sequence, i.e., the time is negligible or can be added to job processing time [14]. However, according to Kopanos et al. [19] and Gupta and Smith [14], in many practical situations these times depend on the execution sequence. Allahverdi et al. [2,3] and [1] provide comprehensive reviews regarding scheduling problems with setup times.

The continuity of machine operations may also be a consideration in the problem. According to Kanet and Sridharan [18], there are situations in which machine idleness is not allowed, because this results in higher costs than the early completion of jobs. However, the same author states that there are cases in which keeping the machine idle is beneficial, even if there is a job available to process. Thus, when there are no restrictions regarding machine idleness, determining the best date to start the execution of each job or insert idle time between jobs may lead to better solutions.

According to the notations employed by Pinedo [26], the scheduling problem studied here, SMSPETP, is represented by $1/s_{jk}/\sum_{j=1}^{n} w'_j E_j + \sum_{j=1}^{n} w''_j T_j$. The case with sequence-independent setup times is denoted by SMSPETP-SIS. Pinedo [26] represents SMSPETP-SIS by $1//\sum_{j=1}^{n} w'_j E_j + \sum_{j=1}^{n} w''_j T_j$. The applicability combined with the difficulty of finding an optimal solution for SMSPETP has motivated the development of efficient algorithms for the resolution of this problem.

Koulamas [20] studied SMSPETP-SIS by considering that a job is anticipated when its execution begins, before the beginning of its time window. This approach is different than that usually found in the literature, in which a job is usually considered to be early if it is completed before the beginning of its time window. Earliness and tardiness penalties are considered to be equal for each job. The author divides the problem into two subproblems: determining the job execution sequence and determining the optimal time for completion of each job in a given sequence (or inserting idle time between jobs in the sequence). To solve the problem of determining the optimal execution time for each job in a given sequence, an algorithm was developed that optimally inserts idle time into the execution sequence. The scheduling problem is solved with adjustments of heuristics that were previously employed in special SMSPETP-SIS cases.

Wan and Yen [31] studied various properties of SMSPETP-SIS to facilitate its resolution. As in the case of Koulamas [20], the authors divided the problem into two subproblems. An optimal timing algorithm was developed to determine the optimal completion date for each job in a given execution sequence. This algorithm, from now on denoted by OTA, is an extension of the algorithms of Davis and Kanet [7], Lee and Choi [21], and Szwarc and Mukhopadhyay [30], to cases with different due dates. Subsequently, a tabu search procedure [12] was proposed, together with the optimal timing algorithm, to solve the JIT scheduling problem.

The study of Gomes Júnior et al. [13] focuses on SMSPETP. In this study, a mixed integer linear programming model is proposed to represent the problem, and a heuristic resolution algorithm based on greedy randomized adaptive search procedures (GRASP) [8], iterated local search (ILS) [22], and variable neighborhood descent (VND) [23] is applied to solve it. For each job sequence generated by the developed heuristic, OTA [31] – adapted to include setup times – is applied to determine the optimal starting time of each job.

Many papers can be found in the literature that use OTA in heuristic procedures to solve SMSPETP (e.g., [28,29], and [25]).

In the present study, an implicit enumeration algorithm denoted IE and an adaptation of the general variable neighborhood search algorithm denoted GVNS [15] are proposed to determine the job execution sequence of SMSPETP. IE is an exact algorithm, which is only valid for solving SMSPETP-SIS, and makes use of theoretical results established for this problem. On the other hand, GVNS is a heuristic algorithm, and can be applied to solve larger SMSPETP instances. A new $O(n^2)$ algorithm for the optimal allocation of idle time, named ITIA, is developed to determine the starting time for each job in a given SMSPETP job sequence. Computational experiments compare the performances of ITIA and OTA [31] when employed by the IE and GVNS algorithms. OTA is the only equivalent algorithm to ITIA found in the literature. The results confirm the superiority of ITIA over OTA. In addition, because no specific exact optimization algorithm for solving SMSPETP-SIS can be found in the literature [16], the IE algorithm is compared with the CPLEX solver applied to the mixed integer linear programming model of Gomes Júnior et al. [13]. The obtained results show that the IE algorithm generates optimal solutions more quickly. Finally, the developed GVNS algorithm is compared with the adaptive genetic algorithm (AGA) [28] for larger instances. The results show that the GVNS algorithm outperforms AGA both in the quality of solutions and processing times.

The remainder of this paper is organized as follows. Section 2 presents a detailed description of SMSPETP and SMSPETP-SIS. The proposed algorithm for the optimal allocation of idle time is presented in Section 3. The GVNS and IE algorithms are presented in Sections 4 and 5, respectively. Section 6 presents and discusses the computational results. Finally, Section 7 presents the conclusions.

2. Characteristics of the addressed problem

The single machine scheduling problem addressed in this study (SMSPETP) has the following characteristics:

(i) A single machine must process a set I of n jobs.
(ii) For each job $x \in I$, there is a processing time P_x and a time window $[E_x, T_x]$ in which the job x should preferably be completed. E_x indicates the earliest due date, and T_x is the tardiest due date.
(iii) If job x is completed before E_x, then there is a cost of α_x per unit of earliness time. In the case that the job is completed after T_x, there is a cost of β_x per unit of tardiness time. Jobs completed within their time windows do not incur costs.
(iv) The machine can perform only one job at a time and once the process is initiated, it cannot be interrupted.
(v) All jobs are available for processing starting from time 0.
(vi) Between two consecutive jobs x and $y \in I$, a setup time of S_{xy} is required. It is assumed that the time for setting up the machine in order to process the first job in the sequence is equal to 0.
(vii) Idle time between the execution of two consecutive jobs is allowed. The time of completion of job $x \in I$ is represented by C_x, whereas the earliness and tardiness times of x are represented by $e_x = \max(0, E_x - C_x)$ and $t_x = \max(0, C_x - T_x)$, respectively.

The special case with sequence-independent setup times (SMSPETP-SIS) is also considered. In this case, the value of S_{xy} is the same for any pair of jobs x and $y \in I$. Therefore, it can be added to the processing time of the respective jobs.

The objective is to determine a job sequence X of I and the starting dates for executing the jobs that minimize the weighted sum of the earliness and tardiness for each job. That is, to minimize the value of

$$f(X) = \sum_{x \in I} (\alpha_x e_x + \beta_x t_x). \quad (1)$$

Table 1
List of notations.

Notations	Descriptions
SMSPETP	Single Machine Scheduling Problem with Distinct Time Windows and Sequence-Dependent Setup Times
SMSPETP-SIS	Single Machine Scheduling Problem with Distinct Time Windows and Sequence-Independent Setup Times
OTA	Optimal Timing Algorithm [31]
ITIA	Idle Time Insertion Algorithm
I	set of n jobs to be executed
X	sequence of the job set I, i.e., $X = (x_1, x_2, \ldots, x_n)$
x_i	ith job in a given sequence X
B	subsequence of jobs, i.e., $B = (x_u, x_{u+1}, \ldots, x_v)$, $1 \leq u \leq v \leq n$
$\|B\|$	cardinality of subsequence B
B_j	jth subsequence of a sequence X
$MC(B)$	marginal cost for shifting jobs from B in a single time unit to the right
$Cost_j(\varphi)$	cost for moving jobs from B_j by φ time units to the right
$last(x_i)$	last job of the subsequence that contains job x_i
GVNS	General Variable Neighborhood Search metaheuristic [15]
$GVNSmax$	stopping criterion of GVNS
N_1, N_2 and $N3$	neighborhoods used in GVNS and VND
$f(X)$	evaluation function of sequence X, $f(X) = \sum_{x \in I}(\alpha_x e_x + \beta_x t_x)$
VND	Variable Neighborhood Descent procedure [23]
$VNDmax$	parameter of VND procedure
IE	Implicit Enumeration algorithm
UB	known upper bound to SMSPETP-SIS
L	list of nodes to be investigated in IE algorithm
δ	node of the search tree in IE algorithm
$seq(\delta)$	subsequence of k jobs stored in node δ, $0 \leq k \leq n$
$nseq(\delta)$	subset of jobs that does not belong to $seq(\delta)$
X_B	sequence of jobs in I that begins with the subsequence B
C_x^X	completion time of the job x in an optimal positioning of X
GVNS/ITIA	GVNS that uses ITIA algorithm
GVNS/OTA	GVNS that uses OTA algorithm
IE/ITIA	IE that uses ITIA algorithm
IE/OTA	IE that uses OTA algorithm
AGA	Adaptive Genetic Algorithm [28]

The resolution of this problem involves two actions, which must be performed in sequence: (i) determine the job processing sequence, i.e., the order in which the jobs must be executed, and (ii) determine the starting date for each job such that the weighted sum of earliness and tardiness for all jobs is minimized.

2.1. Notations

In addition to the basic notations introduced above, Table 1 presents further notations employed in this paper.

3. Idle time allocation

Solving SMSPETP involves the determination of the sequence in which the jobs are executed, as well as the date at which each job in this sequence begins. Previous studies have addressed this problem using heuristics that iteratively generate an execution sequence by priority rules or search procedures. To evaluate each generated sequence, the determination of the optimal starting date for the execution of each job in the sequence is required. This can be achieved by inserting idle times between the executions of consecutive jobs. Therefore, it is essential to use an algorithm that inserts idle times as quickly as possible.

To schedule jobs with different due dates, Fry et al. [10] proposed a mathematical programming formulation that solves the problem of determining optimal dates. Garey et al. [11] presented an $O(n \log n)$ algorithm to determine optimal dates for the problem of minimizing the weighted sum of discrepancies from the preferred due dates. Yano and Kim [32] suggested a dynamic programming algorithm to determine the optimum dates in problems where the penalty per earliness unit for a given job is not greater than the respective penalty per tardiness unit. Szwarc and Mukhopadhyay [30] proposed an algorithm based on the clusters concept, which determines the optimal dates in problems with generic weights for earliness and tardiness penalties. Davis and Kanet [7] and Lee and Choi [21] proposed optimal timing algorithms for problems with the same characteristics. Reviews of scheduling problems involving the insertion of idle times is found in Kanet and Sridharan [18] and Józefowska [17].

Considering problems with distinct time windows, proposed procedures for determining optimal dates are limited to studies by Koulamas [20] and Wan and Yen [31]. In Koulamas [20], an algorithm of polynomial complexity for problems with unweighted earliness and tardiness penalties is presented. This algorithm is an extension of that in the study of Garey et al. [11]. Wan and Yen [31] extended the optimal timing algorithms of Davis and Kanet [7], Lee and Choi [21], and Szwarc and Mukhopadhyay [30] to problems with different time windows and arbitrary weights for earliness and tardiness penalties.

In the following, a new $O(n^2)$ algorithm called the idle time insertion algorithm (ITIA) is proposed for the optimal allocation of idle times. This algorithm is motivated by the study of França Filho [9], which addressed the problem of scheduling in unrelated parallel machines with sequence-dependent setup times, earliness and tardiness penalties, release times, and due dates for each job.

Let $X = (x_1, x_2, \ldots, x_n)$ be a given sequence of the job set I. Thus, x_i, $x_j \in I$ and $x_i \neq x_j$ for every $i,\ j \in \{1, 2, \ldots,\ n\}$. ITIA consists of two steps. First, all jobs are scheduled to start processing as soon as possible, respecting the sequence X and without idle times inserted. Thus, the costs resulting from tardiness are minimized, whereas the costs caused by earliness are at the highest possible level for the sequence X. Second, idle times are inserted between each pair of consecutive jobs, such that the total sum of penalties for earliness and tardiness is reduced.

Let C_x be the completion time of job x. In the first step of ITIA, the execution of the first job of the sequence X is scheduled to start at time 0, i.e., $C_{x_1} = P_{x_1}$. The completion time of the other jobs is given by $C_{x_i} = C_{x_{i-1}} + S_{(x_{i-1})(x_i)} + P_{x_i}$, for $i = 2, 3, \ldots, n$.

In contrast to the algorithm of França Filho [9], the second step of ITIA starts from the last job in the sequence (x_n). From the last to the first job successively, it is verified whether the insertion of idle time is beneficial. If the last job is not early at the end of the first step, then it will not incur a shift to the right, because this would not reduce the associated earliness cost. Otherwise (i.e., if x_n is early), it is shifted φ units to the right, with φ given by

$$\varphi = E_{x_n} - C_{x_n}. \tag{2}$$

A job subsequence $B = (x_u, x_{u+1}, \ldots, x_v)$ with $u \leq v$ forms a *block* in the sequence X if the jobs in B are scheduled consecutively without idle time between them and there is idleness between jobs x_{u-1} and x_u and jobs x_v and x_{v+1}. For the case in which $u = 1$, the idleness between jobs x_{u-1} and x_u is disregarded. Similarly, for the case in which $v = n$ the idleness between jobs x_v and x_{v+1} is disregarded.

The insertion of idle time before jobs x_i, with $i = n-1, n-2, \ldots, 1$, is performed as follows:

- If $C_{x_i} \geq E_{x_i}$, no idle time is inserted before x_i.
- If $C_{x_i} < E_{x_i}$, then it is verified whether a reduction to the earliness of job x_i is beneficial. For this purpose, let $last(x_i)$ be the last job in the block containing x_i. To reduce the earliness of job x_i, all jobs from $B = \{x_i, x_{i+1}, \ldots, last(x_i)\}$ must be shifted to the right. The marginal cost MC of shifting jobs from B a single time unit to the right is given by

$$MC(B) = \sum_{x \in B : C_x \geq T_x} \beta_x - \sum_{x \in B : C_x < E_x} \alpha_x. \tag{3}$$

  In consequence:
  - If $MC(B) \geq 0$, then shifting jobs from the set B to the right is not beneficial, because the increase in tardiness costs will be higher than the reduction in earliness costs.
  - If $MC(B) < 0$, then shifting jobs from set B to the right is beneficial. Moreover, $MC(B)$ will still be negative if the jobs from B are moved up to $\varphi = \min\left(C_{last(x_i)+1} - P_{last(x_i)+1} - S_{(last(x_i))(last(x_i)+1)}, m_1, m_2\right)$ time units to the right, where $m_1 = \min_{x \in B : E_x \leq C_x < T_x}(T_x - C_x)$, $m_2 = \min_{x \in B : C_x < E_x}(E_x - C_x)$, and $last(x_i) + 1$ represents the job scheduled immediately after job $last(x_i)$. This is because the mentioned shift does not change the early/tardy status of the jobs from B. For the case in which $last(x_i) = x_n$, the value of φ is given by $\varphi = \min(m_1, m_2)$. Thus, jobs from B are shifted by φ time unites to the right. Consequently, the following cases may occur:
    - If $C_{last(x_i)} + S_{(last(x_i))(last(x_i)+1)} + P_{last(x_i)+1} = C_{last(x_i)+1}$, then the set B joins the successor block. The set B and the element $last(x_i)$ are updated. Subsequently, the benefit of shifting jobs from the new set B to the right (via $MC(B)$ analysis) is determined.
    - If $C_{last(x_i)} + S_{(last(x_i))(last(x_i)+1)} + P_{last(x_i)+1} < C_{last(x_i)+1}$, then it is necessary to verify whether shifting jobs from set B to the right (via $MC(B)$ analysis) is still beneficial.

The algorithm ends when inserting additional idle time in the sequence X is not beneficial.

The cost incurred by the earliness or tardiness of a job $x \in I$ completed at date C_x can be determined by the following function:

$$g_x(C_x) = \alpha_x \cdot \max\{0, E_x - C_x\} + \beta_x \cdot \max\{0, C_x - T_x\}$$

The function $g_x(C_x + \varphi)$ is a piecewise linear convex function with respect to φ for every $x \in I$, as shown in Wan and Yen [31]. Consider the case that there are l blocks in X, i.e., $X = B_1 \cup B_2 \cup \ldots \cup B_l$. Thus, if jobs from the set B_j are moved φ time units to the right, then the associated cost function will be given by

$$Cost_j(\varphi) = \sum_{x \in B_j} g_x(C_x + \varphi) \qquad \forall j \in \{1, 2, \ldots, l\}, \tag{4}$$

Table 2
Instance to exemplify the application of ITIA procedure.

	Data					Setup			
Jobs	P	E	T	α	β	1	2	3	4
1	3	14	15	2	4	0	2	1	2
2	4	22	24	7	9	1	0	2	3
3	4	9	12	7	8	1	3	0	1
4	3	5	7	1	4	1	2	2	0

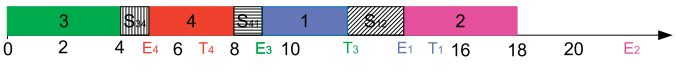

Fig. 1. Initial positioning of the ITIA procedure.

Fig. 2. Positioning after one iteration of the second step of the ITIA procedure.

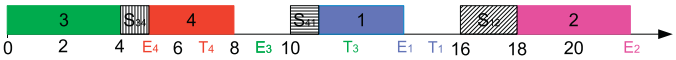

Fig. 3. Positioning after iteration 2 of the second step of the ITIA procedure.

where $C_x + \varphi$ is the new completion date for the execution of job x.

Lemma 1 [5]. *The sum of two piecewise linear convex functions is also a piecewise linear convex function.*

Based on Lemma 1, the following can be shown.

Proposition 1 [31]. *$Cost_j(\varphi)$ is a piecewise linear convex function with respect to φ, for every $j \in \{1, 2, \ldots, l\}$.*

Owing to the piecewise linear convex nature of the cost function, the minimum cost of a block B_j occurs at the extreme points of its function $Cost_j$, i.e., at the beginning or at the end of the time window of one of the jobs in the block. This fact, in conjunction with Proposition 2, ensures that the presented algorithm determines the optimal starting times for the given job sequence.

Proposition 2 [31]. *The total cost of a given job sequence achieves its optimal value if each block B_j in the sequence reaches its minimum point, except that B_1 may have its execution scheduled to start at time zero.*

To illustrate how ITIA works, consider the scheduling problem of four jobs represented in Table 2. This table shows the processing time (P_x), starting date (E_x), ending date (T_x), cost per earliness unit (α_x), and cost per tardiness unit (β_x) of the time window, as well as the setup time relative to each job $x \in I$.

Given the sequence $X = (3, 4, 1, 2)$, all jobs are scheduled to start processing as soon as possible, as shown in Fig. 1. Thus, the sum of penalties resulting from earliness and tardiness is 71 units.

The second step verifies whether inserting idle time before the execution of each job is beneficial. This procedure is performed iteratively, starting with the last job in the sequence ($x_4 = 2$) and ending with the first ($x_1 = 3$). Because inserting idle time before the execution of the job $x_4 = 2$ reduces its earliness ($MC(\{x_4\}) = -7 < 0$), x_4 is shifted four time units to the right (Fig. 2). Following this shift, the sum of earliness and tardiness penalties is equal to 43.

Iteration 2 of the second step of ITIA verifies whether the insertion of idle time before the execution of the job $x_3 = 1$ is beneficial. Because the insertion of time is advantageous ($MC(\{x_3\}) = -2 < 0$), x_3 is shifted two time units to the right (Fig. 3). Following

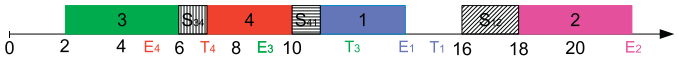

Fig. 4. Positioning after iteration 4 of the second step of the ITIA procedure.

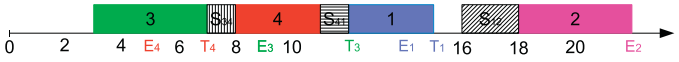

Fig. 5. Optimal positioning for the sequence X.

this shift, the sum of earliness and tardiness penalties is equal to 39.

Iteration 3 of the second step of ITIA verifies whether inserting idle time before the execution of the job $x_2 = 4$ is beneficial. As this insertion of idle time is not beneficial ($MC(\{x_2\}) = 4 \geq 0$), the ITIA procedure goes to iteration 4. Iteration 4 analyzes whether inserting idle time before the execution of the first job in the sequence ($x_1 = 3$) is beneficial. To make this time insertion possible, the job block $(x_1, x_2) = (3, 4)$ must be shifted to the right. Because a shift is beneficial ($MC(\{x_1, x_2\}) = -7 + 4 = -3 < 0$), the block (x_1, x_2) is shifted two time units to the right. Following this shift, the sum of earliness and tardiness penalties is equal to 33. A new block $(x_1, x_2, x_3) = (3, 4, 1)$ is then formed (Fig. 4). At this point, it is necessary to verify whether this new block is in the optimal position. Because shifting this block to the right is beneficial $MC(\{x_1, x_2, x_3\}) = -7 + 4 + 0 = -3 < 0$), it is shifted one time unit to the right (Fig. 5). Following this shift, the sum of earliness and tardiness penalties is equal to 30. Because the inclusion of more idle time before the first job does not improve the solution ($MC(\{x_1, x_2, x_3\}) = -7 + 4 + 4 = 1 \geq 0$), this position is optimal for the sequence X.

3.1. ITIA complexity analysis

Algorithm 1 presents the pseudo-code for ITIA applied to a sequence $X = (x_1, x_2, \ldots, x_n)$ of I.

It is straightforward to see that the initialization of ITIA, lines 1–3 of Algorithm 1, has a computational complexity of $O(n)$. For each $i \in \{n, n-1, \ldots, 2, 1\}$ in the loop in line 4, the computational cost associated to lines 6, 7, and 8 is $O(|B|)$, where $|B|$ is the cardinality of B. The size of the largest possible set B at iteration i is $n - i + 1$ (i.e., $|B| \leq n - i + 1, \ \forall i \in \{n, n-1, \ldots, 2, 1\}$). Therefore, the computational complexity relating to lines 6, 7, and 8 of Algorithm 1 is $O(n - i + 1)$.

It is also easy to see that the computational cost associated with lines 10–22 of the ITIA algorithm is $O(|B|) = O(n - i + 1)$, whereas the cost relating to line 23 is $O(n)$. Thus, if $mc(i)$ denotes the number of times that the "while loop" (lines 9–22) is performed in iteration $i \in \{n, n-1, \ldots, 2, 1\}$ of ITIA (i.e., the number of times that $MC(B) < 0$ at iteration i), then the complexity of the ITIA algorithm is

$$O\left(n + \sum_{i=1}^{n}\left((n-i+1) + \sum_{j=1}^{mc(i)}(n-i+1)\right)\right) = O\left(n^2 + \sum_{i=1}^{n}(n-i+1)\cdot mc(i)\right).$$

Lemma 2. *If $mc(i)$ denotes the number of times that $MC(B) < 0$ occurs in lines 9–22 of iteration $i \in \{n, n-1, \ldots, 2, 1\}$ of the ITIA algorithm, then*

$$\sum_{i=1}^{n} mc(i) \leq 3n - 1.$$

Algorithm 1: $ITIA(n, I, X)$.

1 $C_{x_1} \leftarrow P_{x_1}$;
2 **for** $i = 2, \ldots, n$ **do**
3 $\quad C_{x_i} \leftarrow C_{x_{i-1}} + S_{(x_{i-1})(x_i)} + P_{x_i}$;
end
4 **for** $i = n, n-1, \ldots, 2, 1$ **do**
5 $\quad$ **if** $C_{x_i} < E_{x_i}$ **then**
6 $\quad\quad last(x_i) \leftarrow$ last job of the block that contains x_i;
7 $\quad\quad B \leftarrow \{x_i, x_{i+1}, \ldots, last(x_i)\}$;
8 $\quad\quad MC(B) \leftarrow \sum_{x \in B\,:\, C_x \geq T_x} \beta_x - \sum_{x \in B\,:\, C_x < E_x} \alpha_x$;
9 $\quad\quad$ **while** $MC(B) < 0$ **do**
10 $\quad\quad\quad m_1 \leftarrow \min_{x \in B\,:\, E_x \leq C_x < T_x} (T_x - C_x)$;
11 $\quad\quad\quad m_2 \leftarrow \min_{x \in B\,:\, C_x < E_x} (E_x - C_x)$;
12 $\quad\quad\quad$ **if** $last(x_i) \neq x_n$ **then**
13 $\quad\quad\quad\quad \varphi \leftarrow \min\left(C_{last(x_i)+1} - P_{last(x_i)+1} - S_{(last(x_i))(last(x_i)+1)}, m_1, m_2\right)$;
$\quad\quad\quad$ **else**
14 $\quad\quad\quad\quad \varphi \leftarrow \min(m_1, m_2)$;
$\quad\quad\quad$ **end**
15 $\quad\quad\quad$ **for** $x \in B$ **do**
16 $\quad\quad\quad\quad C_x = C_x + \varphi$;
$\quad\quad\quad$ **end**
17 $\quad\quad\quad$ **if** $C_{last(x_i)} + S_{(last(x_i))(last(x_i)+1)} + P_{last(x_i)+1} = C_{last(x_i)+1}$ **then**
$\quad\quad\quad\quad$ `// B joins with the successor block and it is updated.`
18 $\quad\quad\quad\quad last(x_i) \leftarrow$ last job of the block that contains x_i;
19 $\quad\quad\quad\quad$ **if** $i < n$ **then**
20 $\quad\quad\quad\quad\quad B \leftarrow \{x_i, x_{i+1}, \ldots, last(x_i)\}$;
$\quad\quad\quad\quad$ **else**
21 $\quad\quad\quad\quad\quad B \leftarrow \{x_n\}$;
$\quad\quad\quad\quad$ **end**
$\quad\quad\quad$ **end**
$\quad\quad\quad$ `// MC(B) is recalculated.`
22 $\quad\quad\quad MC(B) \leftarrow \sum_{x \in B\,:\, C_x \geq T_x} \beta_x - \sum_{x \in B\,:\, C_x < E_x} \alpha_x$;
$\quad\quad$ **end**
$\quad$ **end**
end
23 $f \leftarrow \sum_{x \in I} \alpha_x \cdot \max(0, E_x - C_x) + \beta_x \cdot \max(0, C_x - T_x)$;
24 Return f;

Proof. Given that the shifting of the jobs is only performed to the right in the ITIA algorithm, the following are the only two possibilities each time that $MC(B) < 0$:

(i) Block B is shifted until a job is completed at the first limit of its respective time window.
(ii) Block B is shifted until it joins with the successor block and is updated.

Because each job will be at a limit of its time window at most once, it follows that $2n$ is an upper bound on the number of times case (i) occurs (because there are n jobs, and the time window of each job $x \in I$ has two limits, E_x and T_x). On the other hand, once they are together, two jobs do not separate. Therefore, $n - 1$ is an upper bound on the number of times case that (ii) occurs, and $2n + (n - 1) = 3n - 1$ is an upper bound on $\sum_{i=1}^{n} mc(i)$. □

The following is a consequence of Lemma 2:

$$\begin{aligned}\sum_{i=1}^{n}(n-i+1)\cdot mc(i) &= n\cdot mc(1)+(n-1)\cdot mc(2)+\ldots \\ &\quad +2\cdot mc(n-1)+1\cdot mc(n) \\ &= \underbrace{mc(1)+mc(2)+\ldots+mc(n-1)+mc(n)}_{\leq 3\cdot n-1} \\ &\quad +\underbrace{mc(1)+mc(2)+\ldots+mc(n-1)}_{\leq 3\cdot(n-1)-1} \\ &\quad \vdots \\ &\quad +\underbrace{mc(1)+mc(2)}_{\leq 3\cdot 2-1} \\ &\quad +\underbrace{mc(1)}_{\leq 3\cdot 1-1}.\end{aligned}$$

Hence,

$$\sum_{i=1}^{n}(n-i+1)\cdot mc(i) \leq \sum_{i=1}^{n} 3i-1 = \frac{n(3n+1)}{2}.$$

The following proposition summarizes this result.

Proposition 3. *The computational complexity of the ITIA algorithm is* $O(n^2)$.

Similarly, it can be proved that the complexity of the optimal timing algorithm (OTA) of Wan and Yen [31] is also $O(n^2)$.

4. GVNS applied to the considered problem

This section presents the adaptation of the general variable neighborhood search (GVNS) metaheuristic [15] that is proposed in this study to solve SMSPETP. Its pseudo-code is presented in Algorithm 2. *GVNSmax* corresponds to the maximum number of iterations without an improvement in the best known solution, which sets the stopping criterion of the algorithm. In the following subsections, the GVNS algorithm is detailed.

Algorithm 2: GVNS$(I, n, f, N_1, N_2, N_3, GVNSmax, VNDmax)$.

```
X ← InitialSolution(I, n);                    // See Section 4.2
f* ← f(X);                                    // See Section 4.4
Iter ← 0;
while Iter < GVNSmax do
    Iter ← Iter + 1;
    k ← 1;
    while k ≤ 3 do
        Randomly generates a neighbor X' ∈ N_k(X);
        X'' ← VND(X', f, N_1, N_2, N_3, VNDmax);   // See Section 4.5
        if f(X'') < f(X) then
            X ← X'';
            f* ← f(X);
            k ← 1;
            Iter ← 0;
        else
            k ← k + 1;
        end
    end
end
Return f*;
```

4.1. Solution representation

A solution for SMSPETP with n jobs is represented by a sequence X of length n. Each index $i = 1, 2, \ldots, n$ indicates the job to be executed at position i of X. For example, in the sequence $X = (5, 1, 2, 6, 4, 3)$, job 5 is the first to be executed, and job 3 is the last.

4.2. Initial solution

An initial solution for SMSPETP is constructed by applying the earliest due date (EDD) heuristic. EDD is a greedy constructive heuristic, often applied in the literature to scheduling problems with distinct due dates [26]. The proposed construction begins with an empty execution subsequence (i.e., no jobs are in the sequence). Iteratively, the job with the earliest starting date for its time window of those not yet sequenced is inserted at the end of the current subsequence. Ties are broken randomly. The construction procedure is stopped when no more jobs lie outside of the execution sequence.

4.3. Neighborhood of a solution

To explore the solution space, three types of movements are considered:

(i) pairwise interchange;
(ii) one job reallocation;
(iii) subsequence reallocation.

These movements define the neighborhoods N_1, N_2, and N_3, respectively, described below.

4.3.1. Neighborhood N_1

An example of a neighbor of $X = (5, 3, 2, 1, 4, 6)$ in the neighborhood N_1 is $X' = (5, 4, 2, 1, 3, 6)$. Note that X' is obtained from X by swapping the execution positions of jobs 3 and 4.

For a given sequence of n jobs, the position of each job can be swapped with that of the remaining $n-1$ jobs. On the other hand, swapping the position of the ith job in the sequence with that of the jth job is equivalent to swapping the position of the jth job with that of the ith job, for every $i, j \in \{1, 2, \ldots, n\}$. Therefore, there are $n(n-1)/2$ different neighbors in respect to the neighborhood N_1.

4.3.2. Neighborhood N_2

The solution $X' = (5, 2, 1, 4, 3, 6)$ is an example of neighbor of $X = (5, 3, 2, 1, 4, 6)$ in the neighborhood N_2. In fact, X' is obtained from X by reallocating job 3 (which is in the second position in X) to the fifth position.

Given a sequence of n jobs, each one may be reallocated to $n-1$ distinct positions. In addition, reallocating the job from position i to position $i+1$ is equivalent to reallocating the job in position $i+1$ to position i, for every $i \in \{1, 2, \ldots, n-1\}$. Therefore, there are $(n-1)^2$ distinct neighbors in the neighborhood N_2.

4.3.3. Neighborhood N_3

The solution $X' = (1, 4, 5, 3, 2, 6)$ is in neighborhood the N_3 of $X = (5, 3, 2, 1, 4, 6)$. This solution is obtained by reallocating the subsequence $< 5, 3, 2 >$ with the three consecutive jobs in the first three positions of X to after job 4.

Given a sequence of n jobs, there are $n-k+1$ distinct subsequences of k jobs for every $k \in \{1, 2, \cdots, n\}$. Each of these subsequences may be reallocated to $n-k$ distinct positions in the sequence. In addition, reallocating a subsequence with k_1 jobs to k_2 successor positions is equivalent to reallocating a subsequence

with k_2 jobs to k_1 predecessor positions, for every k_1 and $k_2 \in \{1, 2, \ldots, n\}$ and $k_1 + k_2 \leq n$. Thus, there are $(n^3 - n)/6$ distinct neighbors with respect to the neighborhood N_3.

In order to prevent returning to previously analyzed neighbors and to reduce the computational cost, all previous moves are avoided in the computational implementation of the GVNS algorithm. Furthermore, the chosen order for the exploration of neighborhoods explores increasingly distant neighborhoods of the current incumbent solution in terms of computational complexity, as proposed by Hansen et al. [15].

4.4. Evaluation of a solution

Any job sequence can produce a feasible solution to SMSPETP by means of an optimal positioning algorithm (e.g., the ITIA or OTA algorithms – see Section 3).

4.5. Variable neighborhood descent

The local search used in the GVNS algorithm is performed using the variable neighborhood descent (VND) procedure [23]. In this study, VND uses the following sequence of local searches:

- LS1: Random descent using neighborhood N_1.
- LS2: Random descent using neighborhood N_2.
- LS3: Random descent using neighborhood N_3.

In the first local search (LS1), two jobs are randomly chosen, and their positions in the sequence are swapped. If the new sequence gives an improved solution, then it is accepted and becomes the current solution. Otherwise, another random swap in the current solution is evaluated. LS1 ends when *VNDmax* successive swaps occur without an improvement in the current solution, where *VNDmax* is a parameter of the procedure. When LS1 finishes, the next local search (LS2) is applied.

In the local search LS2, a job of the sequence and a new position for this job are randomly chosen. If the new sequence gives a better solution, then it becomes the current solution, and VND switches back to the local search LS1. Otherwise, another random reallocation in the current solution is evaluated. LS2 is interrupted when *VNDmax* successive reallocations occur without an improvement in the current solution. In this case, the next local search (LS3) is conducted.

In the local search LS3, a subsequence of jobs of the sequence and a new position for this subsequence are randomly chosen. The subsequence size is also chosen randomly in the interval $[1, n-1]$. If the new sequence gives an improved solution, then it becomes the current solution, and VND switches back to the local search LS1. Otherwise, another random subsequence reallocation in the current solution is evaluated. LS3 is interrupted after *VNDmax* successive subsequence reallocations occur without an improvement in the current solution. In this case, VND is stopped, and the best solution is returned.

Once again, the three neighborhoods of the incumbent solution are explored in order to visit increasingly distant neighbors.

5. Implicit enumeration

In this section, an exact algorithm based on implicit enumeration (IE) is proposed to solve SMSPETP-SIS. According to Janiak et al. [16], there is no specific exact optimization algorithm for solving SMSPETP-SIS. Let I be the set of n jobs that must be scheduled. Let UB be a known upper bound on the problem, which can be obtained heuristically. During the enumeration process, a node δ corresponds to a structure that stores a subsequence of k jobs ($0 \leq k \leq n$) as well as the set of $n-k$ jobs that are outside of the subsequence, which are represented by $seq(\delta)$ and $nseq(\delta)$, respectively. Let L be the list of nodes to be investigated. The job sequence X is evaluated by the function f given by Eq. (1), as in Section 4.4.

Let δ_0 be the node such that $seq(\delta_0) = \varnothing$ and $nseq(\delta_0) = I$. The IE algorithm is initialized by inserting the node δ_0 as the only node of L. Thereafter, each IE iteration consists of the following steps:

Step 1: Let δ be the last node inserted into L.
Step 2: The node δ is removed from L.
Step 3: If $nseq(\delta) = \varnothing$, then the subsequence $seq(\delta)$ corresponds to a complete solution. In addition, if $f(seq(\delta)) < UB$ then the value of UB is updated to $UB = f(seq(\delta))$.
Step 4: If $nseq(\delta) \neq \varnothing$, then for each job $x \in nseq(\delta)$ it is verified whether the subsequence $seq(\delta) \cup \{x\}$ obtained by inserting the job x at the end of the subsequence $seq(\delta)$ violates the optimality conditions (presented in Section 5.1). If these conditions are not violated, then the node δ_{son} is inserted into L, where $seq(\delta_{son}) = seq(\delta) \cup \{x\}$ and $nseq(\delta_{son}) = nseq(\delta) \setminus \{x\}$.

The IE algorithm is stopped when $L = \varnothing$, and the final value of UB corresponds to the optimal solution of the problem.

5.1. Optimality conditions

For ease of notation, let $seq(\delta) = B = (x_1, x_2, \cdots, x_u)$ with $1 \leq u \leq n$ denote a subsequence of jobs in I. An *optimal positioning of a subsequence B* is an optimal scheduling of the jobs (in B) where the order of the jobs is conserved. Furthermore, let X_B denote a sequence of all jobs in I that begins with the subsequence B.

Proposition 4. *If $f(B) > UB$, where UB is a given upper bound on the problem, then there is no optimal scheduling of jobs in I that contains the subsequence B.*

Proof. This result is straightforward, because the inclusion of jobs in B does not decrease the total cost of the jobs in B. □

Lemma 3. *If C_x^B is the completion time of the job x (with x in B) in an optimal positioning of the subsequence B, then there is an optimal positioning of X_B in which x is completed on time, i.e., $C_x^{X_B} \leq C_x^B$.*

Proof. Shifting x to the right does not decrease the total cost of the jobs in B, and moreover, this also does not reduce the sum of the penalties associated with the jobs outside of B. On the other hand, although shifting x to the left does not reduce the cost of B, there is a possibility that the sum of the penalties associated to the jobs outside of B is reduced, so that the cost of X_B is also reduced. Therefore, there exists an optimal positioning of X_B in which $C_x^{X_B} \leq C_x^B$. □

Corollary 1. *If there is an optimal positioning of B in which all jobs are performed consecutively without idle times inserted and the first job starts at time 0, then there is an optimal positioning of X_B in which $C_x^{X_B} = C_x^B$, where $C_x^{X_B}$ and C_x^B are the completion times of x in X_B and B, respectively.*

Proof. Note that in this case, C_x^B is given by the sum of the processing times of the jobs sequenced before x. Therefore, the proof differs from that of Lemma 3 only by the fact that it is now not possible to shift the job x to the left. □

Proposition 5. *If there are two consecutive jobs x and y in B such that $\alpha_y P_x < \alpha_x P_y$ and $C_x^B + P_y = C_y^B \leq \min(E_x, E_y)$ in an optimal positioning of B, then there is no optimal scheduling of jobs in I that begins with the subsequence B.*

Proof. Suppose that there is an optimal sequence X_B. By Lemma 3, there is an optimal completion time of job y in an optimal po-

sitioning of X_B such that $C_y^{X_B} \leq \min(E_x, E_y)$. Thus, the amount of penalties associated to the jobs x and y is given by

$$\alpha_x \cdot (E_x - C_x^{X_B}) + \alpha_y \cdot (E_y - C_y^{X_B}),$$

i.e.,

$$\alpha_x \cdot (E_x - C_y^{X_B} + P_y) + \alpha_y \cdot (E_y - C_y^{X_B}). \quad (5)$$

Let X'_B be the sequence obtained from X_B by just swapping the processing order of the jobs x and y. Then, there exists a positioning of X'_B in which $C_x^{X'_B} = C_y^{X'_B} + P_x$, $C_x^{X'_B} = C_y^{X_B}$, and $C_z^{X'_B} = C_z^{X_B}$, $\forall z \in I \setminus \{x, y\}$. Therefore, the amount of penalties associated to the jobs x and y is given by

$$\alpha_x \cdot (E_x - C_x^{X'_B}) + \alpha_y \cdot (E_y - C_y^{X'_B}),$$

i.e.,

$$\alpha_x \cdot (E_x - C_y^{X_B}) + \alpha_y \cdot (E_y - C_y^{X_B} + P_x). \quad (6)$$

As the completion times of the remaining jobs are the same in the positioning of X_B and X'_B, $f(X_B)$ and $f(X'_B)$ are such that

$$f(X_B) - f(X'_B) = (5) - (6) = \alpha_x P_y - \alpha_y P_x > 0.$$

This contradicts the fact that X_B is an optimal sequence of the jobs in I. □

Proposition 6. *If there exists an optimal positioning of B in which all jobs are performed consecutively without idle times inserted, the first job starts at time 0, and there are two adjacent jobs x and y such that* $cost_1 > cost_2$ *where*

$$cost_1 = \alpha_x e_x + \beta_x t_x + \alpha_y e_y + \beta_y t_y \text{ and}$$
$$cost_2 = \alpha_x \max(0, E_x - C_x^B - P_y) + \beta_x \max(0, C_x^B + P_y - T_x) + \alpha_y \max(0, E_y - C_y^B + P_x) + \beta_y \max(0, C_y^B - P_x - T_y),$$

then there is no optimal scheduling of jobs in I that begins with the subsequence B.

Proof. Suppose that there is an optimal sequence X_B. By Corollary 1, there are optimal completion times for the jobs x and y such that $C_x^{X_B} = C_x^B$ and $C_y^{X_B} = C_y^B$. Consequently, the amount of penalties associated to the jobs x and y is given by $cost_1$. Let X'_B be the sequence obtained from X_B by just swapping the processing order of the jobs x and y. Then, there is a positioning of X'_B in which $C_z^{X'_B} = C_z^{X_B}$, $\forall z \in I \setminus \{x, y\}$. It is easy to see that the amount of penalties associated to the jobs x and y is given by $cost_2$ in this positioning of X'_B. Thus, $f(X_B)$ and $f(X'_B)$ are such that

$$f(X_B) - f(X'_B) = cost_1 - cost_2 > 0.$$

This contradicts the fact that X_B is an optimal sequence of the jobs in I. □

To verify whether a given subsequence of jobs $B = seq(\delta) = (x_1, x_2, \cdots, x_u) \in \delta$ with $1 \leq u \leq n$ can be contained in an optimal sequence (i.e., if it does not violate the optimality conditions), the value of $f(seq(\delta))$ is calculated, and the completion times of the jobs in $seq(\delta)$ are determined. This procedure is performed by applying ITIA (see Section 3) in $seq(\delta)$.

Based on the Propositions 4, 5, and 6, the following corollary highlights the above results.

Corollary 2. *The subsequence* $seq(\delta)$ *violates the condition of optimality if one of the following cases occurs:*

(i) $f(seq(\delta)) > UB$.
(ii) *There are two consecutive jobs x and y in* $seq(\delta)$, *such that* $\alpha_y P_x < \alpha_x P_y$ *and* $C_x^{seq(\delta)} + P_y = C_y^{seq(\delta)} \leq \min(E_x, E_y)$.
(iii) *All jobs in* $seq(\delta)$ *are performed consecutively without idle times inserted, the first job starts at time 0 and there are two adjacent jobs x and y that satisfy the condition of Proposition 6.*

Algorithm 3: IE(I, n, $f(.)$, UB).

```
seq(δ0) ← ∅;
nseq(δ0) ← I;
L ← {δ0};
while L ≠ ∅ do
    δ ← the last node inserted in L;
    L ← L \ {δ};
    if nseq(δ) = ∅ then
        if f(seq(δ)) < UB then
            UB ← f(seq(δ));
        end
    else
        for j ∈ nseq(δ) do
            if the subsequence seq(δ) ∪ {j}
            does not violate the optimality condition then
                seq(δson) ← seq(δ) ∪ {j};
                nseq(δson) ← nseq(δ) \ {j};
                L ← L ∪ {δson};
            end
        end
    end
end
Return UB;
```

Algorithm 3 describes the IE procedure applied to SMSPETP-SIS. In this procedure, the entry UB corresponds to a known upper bound for the problem. This value is updated using the IE procedure, so that the returned value UB is the optimal solution for the problem. In the present work, the initial upper bound is provided by the GVNS algorithm, as presented in Section 4.

6. Computational results

The ITIA, GVNS, and IE algorithms presented in Sections 3, 4, and 5, respectively, were implemented using the C++ language with NetBeans IDE 7.4 as the compiler.

The experiments were performed on a computer with Intel ® Core ™i7-3632QM CPU, 2.20 GHz, 8GB of RAM, and the Ubuntu 13.10 operating system. Although the processor of this device has more than one core, the algorithms are not optimized for multiprocessing.

The instances used to test the algorithms are described in Section 6.1. In Section 6.2, the results associated to the application of GVNS, the IE algorithm, and the CPLEX solver on instances of SMSPETP-SIS containing up to 20 jobs are presented. The algorithms ITIA and OTA are also compared. OTA is the only algorithm found in the literature that is able to determine the optimal completion date for each job in a given execution sequence of SMSPETP. On the other hand, in Section 6.3 the best heuristic algorithm from Section 6.2, namely GVNS using ITIA, is compared with an algorithm from the literature.

6.1. Instance description

In order to evaluate the algorithms GVNS, ITIA, and IE, instances were generated using the methodology of [31] and [27]. This methodology is described below.

For each job $x \in I$, the processing time P_x, cost per tardiness unit β_x, and cost per earliness unit α_x are randomly generated integers within the intervals [1, 40], [1, 10], and [1, β_x], respectively. The time window center of job x is a random integer number within the interval $[(1 - TF - \frac{RDD}{2})TPT, (1 - TF + \frac{RDD}{2})TPT]$, where:

- TPT is the total processing time of all jobs.

Table 3
Comparison of the results obtained with configurations GVNS/OTA, IE/OTA, GVNS/ITIA, IE/ITIA, and CPLEX (average times in seconds over 16 instances).

#	OTA		ITIA		CPLEX
Jobs	GVNS	IE	GVNS	IE	
08	0.08	0.02	0.02	0.01	2.62
09	0.14	0.09	0.02	0.04	18.91
10	0.22	0.44	0.04	0.18	183.67
11	0.35	2.61	0.05	0.97	2085.10
12	0.54	19.45	0.07	6.12	–
13	0.76	212.35	0.09	66.76	–
14	1.07	1126.28	0.13	277.56	–
15	1.40	6594.25	0.19	1799.71	–
16	2.07	–	0.23	–	–
17	2.80	–	0.29	–	–
18	3.47	–	0.38	–	–
19	4.56	–	0.43	–	–
20	6.33	–	0.56	–	–

- *TF* is the tardiness factor.
- *RDD* is the relative range of the time windows.

The time window width is an integer number that is randomly selected in the interval $[0, \frac{TPT}{n}]$, where n is the number of jobs to be scheduled. For any distinct jobs $x, y \in I$, the setup time S_{xy} is an integer number that is randomly selected in the interval [5, 15]. The setup times are not necessarily symmetrical. Sets of instances with 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 30, 40, 50, and 75 jobs were generated. The values 0.1, 0.3, 0.5, and 0.8 were used for *TF*, and 0.4, 0.7, 1.0, and 1.3 for *RDD*. Thus, there are 16 instances in each set, totaling 272 instances. Whenever the setup times for a generated instance did not satisfy the triangle inequality given by

$$S_{xy} \leq S_{xz} + P_z + S_{zy}, \quad \forall\, x, y, z \in I, \quad x \neq y, \quad x \neq z \text{ and } y \neq z \qquad (7)$$

the instance was discarded, and another was generated with the same values for *TF* and *RDD*. Therefore, all instances from the present database satisfy the triangle inequality in relation to the setup times. These instances, and all of the computational results, are available at http://www.decom.ufop.br/prof/marcone/projects/SMSPETP.html.

6.2. Results for instances of up to 20 jobs for SMSPETP-SIS

This section presents the computational results associated to the application of the GVNS and IE algorithms, as well as the CPLEX solver, for solving SMSPETP-SIS. An initial upper bound on the objective function value of the problem used in the IE algorithm is provided by the GVNS procedure. On the other hand, the CPLEX solver is applied to the mathematical programming model of Gomes Júnior et al. [13]. This model was implemented with the C++ Concert Technology of the IBM ILOG CPLEX Optimization Studio 12.5.1 optimizer. The solver was configured to use only one thread. Other parameters remained unchanged.

The instances used in this set of tests each involve up to 20 jobs, on account of the prohibitive computational cost required by the exact algorithms for solving larger instances. Because the IE algorithm is only applied to solve SMSPETP-SIS, the setup times of each instance were disregarded in this set of tests.

The GVNS parameters were empirically set to the values $GVNSmax = VNDmax = 4n$, where n is the total number of jobs to be scheduled. The effect of these parameters on the GVNS algorithm is presented in Section 6.3. Each job sequence generated by the GVNS is evaluated by both the ITIA and OTA [31] algorithms in order to compare the efficiency of these procedures. Considering the stochastic nature of GVNS, each instance is solved 30 times.

Table 3 presents the average times obtained by the application of the GVNS and IE algorithms and the CPLEX solver in instances of up to 20 jobs. In this table, the first column indicates the number of jobs in each set of instances. The second and third columns show the mean times, in seconds, required for OTA to evaluate each job sequence generated by the GVNS and IE algorithms, respectively. Similarly, the fourth and fifth columns show the required computational times when using the ITIA algorithm. Finally, the last column presents the mean time required by CPLEX to solve the instances of each set. For instances with 12 jobs or more, the run time using the CPLEX solver is prohibitive. Similar behavior is observed when applying the IE algorithm to solve instances with more than 15 jobs. Therefore, Table 3 does not include these results.

According to Table 3, the average time required by GVNS with ITIA (GVNS/ITIA) is always lower than the respective average time required by GVNS with OTA (GVNS/OTA). For example, for the set of 20 jobs the average time required by GVNS/OTA is almost 11 times higher than that required by GVNS/ITIA. For instances with up to 11 jobs, GVNS is able to determine the optimal solutions obtained by CPLEX within significantly lower computational times, regardless of the algorithm used to evaluate the generated sequences. For the instance set of 11 jobs, GVNS/ITIA requires an average time of 0.05 s, and CPLEX requires 2085.1 s.

Table 3 also indicates that the average times required by CPLEX are significantly higher than those required by IE. For the instance set of 11 jobs, IE/ITIA requires less than 1 s on average, whereas CPLEX requires 2085.1 s. The average times required by IE/ITIA are always lower than those required by IE/OTA. For example, for the set of 14 jobs the average time required by IE/OTA is more than four times greater than that of IE/ITIA.

If the processing time had been limited to one hour, then IE/OTA would have been unable to solve one of the instances with 14 jobs and six of the instances with 15 jobs. Under the same conditions, only three of the instances with 15 jobs would not have been solved by IE/ITIA.

Although GVNS is a heuristic procedure, it always determines the same solution in all of the runs for each instance of SMSPETP-SIS involving up to 20 jobs. Furthermore, GVNS finds the optimal solutions for all instances where such solutions are known (i.e., instances with up to 15 jobs).

6.3. Results for instances of up to 75 jobs for SMSPETP

For instances of SMSPETP involving between eight and 75 jobs, the GVNS/ITIA algorithm is compared with the best version of the AGA algorithm from Ribeiro et al. [28]. GVNS is employed with ITIA because the ITIA algorithm is faster than OTA in determining the optimal starting dates of the job sequence, according to Section 6.2.

For the comparison with the AGA algorithm, the parameters *GVNSmax* and *VNDmax* of the GVNS/ITIA algorithm are executed with the values $GVNSmax = VNDmax = 3n$, $GVNSmax = VNDmax = 4n$, and $GVNSmax = VNDmax = 5n$, where n is the number of jobs to be scheduled. These parameters characterize the stopping criteria of GVNS and VND, respectively.

In Section 6.3.1, statistical tests are presented to compare GVNS with AGA. In Sections 6.3.2 to 6.3.4, the results achieved by these algorithms are compared. In Section 6.3.2, the number of times that AGA achieved a better performance than GVNS is described. In Section 6.3.3, the number of times that GVNS obtained a better performance than AGA is considered. In Section 6.3.4, these algorithms are compared in relation to the relative average deviation (RAD) metric. Finally, in Section 6.3.5 the average times required by these algorithms are compared.

6.3.1. Hypothesis tests

Considering the stochastic nature of the algorithms, each one is applied 30 times to each instance. On the other hand, instances

with 75 jobs are only solved 10 times per algorithm, on account of the high computational cost.

For each instance, the results obtained by GVNS with the three parameter values are compared with those found by AGA. The comparison is performed by using two one-way hypothesis tests with a significance level of $\gamma = 0.05$. The first test is a parametric hypothesis test for two independent samples [24], whereas the second is a permutation test [6]. Permutation tests are non-parametric statistical methods, i.e., they do not require that samples come from populations with normal distributions [24].

For the two algorithms A_1 and A_2, the following hypotheses (null and alternative) are formulated to compare the averages of their solutions on a given instance:

- Null hypothesis (H_0): The averages of the solutions obtained using the algorithms A_1 and A_2 are equal.
- Alternative hypothesis (H_1): The average of the solutions obtained using algorithm A_1 is lower than the average of solutions obtained using algorithm A_2.

Therefore, the null hypothesis states that there is no significant evidence that the average of the solutions obtained by algorithm A_1 is lower than the average of the solutions obtained by algorithm A_2. Conversely, rejecting the null hypothesis and accepting the alternative hypothesis means that within the adopted significance level, there is significant evidence that the solutions obtained by algorithm A_1 are better on average than the solutions obtained by algorithm A_2.

The permutation test is based on Carrano et al. [6], and is described in Algorithm 4. In this algorithm, $A_1 = (a_1, a_2, \ldots, a_{n_1})$ and $A_2 = (b_1, b_2, \ldots, b_{n_2})$ are vectors that store the solutions obtained by algorithms A_1 and A_2, respectively, as applied to a given instance. The significance level is γ, with $0 < \gamma < 1$.

Algorithm 4: Permutation test(A_1, A_2, γ).

$d \leftarrow \frac{1}{n_2}\sum_{i=1}^{n_2} b_i - \frac{1}{n_1}\sum_{i=1}^{n_1} a_i$;
$V \leftarrow (a_1, a_2, \ldots, a_{n_1}, b_1, b_2, \ldots, b_{n_2})$;
for $i = 1, 2, \ldots, 500$ **do**
  $V' \leftarrow$ the vector V shuffled;
  $w_i \leftarrow \frac{1}{n_2}\sum_{j=1}^{n_2} v'_j - \frac{1}{n_1}\sum_{j=n_2+1}^{n_2+n_1} v'_j$;
end
$p_{value}(d) \leftarrow P(W < d)$; `// Probability that W is smaller than d.`
if $p_{value}(d) \geq 1 - \gamma$ **then**
  reject H_0 and accept H_1;
end

6.3.2. AGA × GVNS

First, in order to verify the number of times that AGA achieved a better performance than GVNS the hypothesis tests are performed with the combination of algorithms given by AGA × GVNS. The rejection of the null hypothesis with this combination indicates that the average solution of AGA is better than the average solution of GVNS (with a significance level of 0.05).

In Table 4, the columns "Parametric Test" and "Permutation Test" present a summary of the results obtained by applying these hypothesis tests to the solutions of each instance. The first column indicates the number of jobs in each set of instances. The columns "GVNS/3n, " "GVNS/4n, " and "GVNS/5n" indicates the number of times that the null hypothesis is rejected when applying the corresponding hypothesis tests with the GVNS parameter values set as 3n, 4n, and 5n, respectively.

Table 4
Number of times AGA is better than GVNS.

#	Parametric test			Permutation test		
Jobs	GVNS/3n	GVNS/4n	GVNS/5n	GVNS/3n	GVNS/4n	GVNS/5n
08	0	0	0	0	0	0
09	0	0	0	0	0	0
10	0	0	0	0	0	0
11	0	0	0	0	0	0
12	1	0	0	0	0	0
13	0	0	0	0	0	0
14	2	0	0	2	0	0
15	1	1	0	1	0	0
16	1	0	0	0	0	0
17	2	1	0	2	0	0
18	0	0	0	0	0	0
19	0	0	0	0	0	0
20	3	1	1	3	1	0
30	7	3	1	8	2	1
40	8	0	0	8	0	0
50	6	1	0	5	1	0
75	1	0	0	2	0	0
Total	32	7	2	31	4	1

According to Table 4, the hypothesis tests (the permutation and parametric tests) obtained similar results. Regardless of the parameter values used in GVNS, the null hypothesis is not rejected in any of the comparisons involving instances of up to 11 jobs. For the other sets of instances, as the GVNS parameter values increase the number of times that the average solution obtained by AGA is better than that obtained by GVNS decreases, in both hypothesis tests. For example, in the 16 instances with 50 jobs, the parametric test indicates that AGA is better than GVNS/3n in six instances, and better than GVNS/4n in just a single case. For instances with 75 jobs, when AGA is compared with GVNS/3n the permutation test indicates that the null hypothesis should be rejected in only two of the 16 instances. With the GVNS parameters fixed to 4n or 5n, there is no significant evidence that the average solutions of AGA are better than the corresponding average solutions of GVNS.

Finally, when the GVNS parameters are fixed to 5n, the parametric and permutation tests indicate that the null hypothesis should be rejected a total of two and one times, respectively, in relation to all instances of all sets.

6.3.3. GVNS × AGA

The hypothesis (parametric and permutation) tests are used to verify the number of times that GVNS obtained a better average solution than AGA. These tests are conducted using a combination of algorithms in the form GVNS × AGA. Thus, rejection of the null hypothesis indicates that there is significant evidence that the average solution of GVNS is better than that of AGA (with a significance level of 0.05).

The columns "Parametric Test" and "Permutation Test" of Table 5 summarize the results when the hypothesis tests are applied to the solutions of each instance. Note that in this table, all columns have the same meanings as those in Table 4.

Table 5 shows that the hypothesis (parametric and permutation) tests return similar results. For the instances with 11 jobs, the parametric hypothesis test indicates that GVNS obtains a better average solution than AGA for one instance when the GVNS parameter values are set to 4n or 5n. Similar behavior can be observed in the instance sets with 13, 15, 16, and 17 jobs, respectively. For instance sets with less than 20 jobs, the permutation test rejects the null hypothesis for only a single instance (with 13 jobs and GVNS parameter values set to 4n or 5n). In other instances, as the GVNS parameter values increase, the number of times that both hypothesis tests show that the average solution of GVNS is better than that of AGA increases. For the set of 16 instances with 75

Table 5
Number of times that GVNS is better than AGA.

#	Parametric test			Permutation test		
Jobs	GVNS/3*n*	GVNS/4*n*	GVNS/5*n*	GVNS/3*n*	GVNS/4*n*	GVNS/5*n*
08	0	0	0	0	0	0
09	0	0	0	0	0	0
10	0	0	0	0	0	0
11	0	1	1	0	0	0
12	0	0	0	0	0	0
13	0	1	1	0	1	1
14	0	0	0	0	0	0
15	0	0	1	0	0	0
16	0	1	1	0	0	0
17	1	1	1	0	0	0
18	0	0	0	0	0	0
19	0	0	0	0	0	0
20	1	3	3	0	3	4
30	0	2	7	0	2	6
40	0	2	9	0	2	8
50	0	6	10	0	6	10
75	0	4	11	0	4	11
Total	2	21	45	0	18	40

jobs each, both of the hypothesis tests suggest that the average solution of GVNS with parameter values set to 4*n* is better than that of AGA in four instances, and that the average solution of GVNS with parameter values set to 5*n* is better than that of AGA for 11 instances. Finally, GVNS/5*n* obtains a better average solution than AGA in a total of 45 and 40 instances according to the parametric and permutation tests, respectively.

The comparisons of results presented in Tables 4 and 5 indicate that the hypothesis (parametric and permutation) tests suggest that AGA is more often better than GVNS/3*n*. However, when considering GVNS/4*n* or GVNS/5*n*, this scenario is reversed. The most significant difference occurs when the GVNS parameter values are set to 5*n*. In this case, the parametric hypothesis test indicates that AGA determined better average solutions than GVNS in two instances, whereas the permutation test indicates the same for only one instance. The same hypothesis tests suggest that GVNS/5*n* is better than AGA in 45 and 40 instances for the parametric test and permutation tests, respectively.

6.3.4. Analysis of the relative average deviation

In order to determine which of the algorithms provides a better quality of solutions, the hypothesis (parametric and permutation) tests are also applied using the relative average deviation (*RAD*) metric. Given an instance i and an algorithm A, the *RAD* of solutions of A for the instance i, denoted RAD_i^A, is calculated as follows:

$$RAD_i^A = \frac{\overline{f_i^A} - f_i^*}{f_i^*}. \quad (8)$$

In this equation, $\overline{f_i^A}$ represents the average value of the solutions found by algorithm A for the instance i, and f_i^* is the value of the best known solution for this instance.

For the considered instances, the value adopted for f_i^* corresponds to the best solution found for the instance i using the AGA algorithm of Ribeiro et al. [28]. Note that the lower the value for RAD_i^A, the better the quality of the solutions obtained by algorithm A for instance i.

The following hypotheses (null and alternative) are developed to compare the average *RAD* values of the GVNS and AGA algorithms:

- Null hypothesis (H_0): The average *RAD* of GVNS is equal to the average *RAD* of AGA.
- Alternative hypothesis (H_1): The average *RAD* of GVNS is lower than the average *RAD* of AGA.

Table 6
Results of the hypothesis tests with the *RAD* metric.

Comparison	Conclusion	
	Parametric test	Permutation test
GVNS/3*n* × AGA	No reject H_0	No reject H_0
GVNS/4*n* × AGA	No reject H_0	Reject H_0
GVNS/5*n* × AGA	Reject H_0	Reject H_0

Table 7
Average times required by the AGA and GVNS algorithms (in seconds).

#	Average time			
Jobs	GVNS/3*n*	GVNS/4*n*	GVNS/5*n*	AGA
08	0.01	0.01	0.02	0.54
09	0.01	0.02	0.03	0.67
10	0.02	0.03	0.05	0.84
11	0.03	0.04	0.07	1.06
12	0.04	0.06	0.10	1.43
13	0.06	0.09	0.13	1.87
14	0.08	0.13	0.19	2.18
15	0.11	0.18	0.26	2.95
16	0.13	0.22	0.32	3.90
17	0.17	0.27	0.42	4.79
18	0.21	0.33	0.50	5.88
19	0.26	0.43	0.63	6.96
20	0.36	0.60	0.90	9.06
30	2.25	3.63	5.15	57.95
40	8.87	14.55	20.49	215.94
50	27.07	42.76	61.62	655.92
75	195.35	306.33	461.46	5213.14

Based on this metric, accepting the null hypothesis means that there is no significant evidence that the average of the solutions found by GVNS is better than the average of those found by AGA. In contrast, rejecting the null hypothesis and accepting the alternative hypothesis means that there is sufficient evidence to state that the average solution obtained by GVNS is better than that obtained by AGA (with a significance level of 0.05).

Table 6 presents the conclusions of the hypothesis (parametric and permutation) tests with the *RAD* metric. GVNS/3*n*, GVNS/4*n*, and GVNS/5*n* represent the applications of GVNS with the parameter values set to 3*n*, 4*n*, and 5*n*, respectively. This table shows that the parametric hypothesis test does not reject the null hypothesis for GVNS/3*n* and GVNS/4*n*. Conversely, these tests suggest that the average solution for GVNS/5*n* is of a higher quality than that of AGA. The permutation test only does not reject H_0 when the GVNS parameter values are set to 3*n*. This indicates that according to this test, when the GVNS parameter values are set to 4*n* or 5*n*, GVNS obtains a better average solution than AGA.

6.3.5. Comparisons in relation to required times

Table 7 presents the average times required (in seconds) by GVNS and AGA for each set of instances. In this table, the first column indicates the number of jobs for each instance set. The averages are relative to the 30 applications of the algorithms for each instance. The values obtained with the GVNS parameter values set to 3*n*, 4*n*, and 5*n* are presented.

Table 7 indicates that the average time required by GVNS is directly related to the parameter values of the algorithm. This occurs because the average time increases whenever the GVNS parameter values increase. A comparison of the average times required by AGA with the corresponding average times required by GVNS shows that the time required by AGA is significantly higher, regardless of the parameter values used for GVNS. Although GVNS/5*n* required a higher computational time than other settings of GVNS,

AGA required average times that were still over 10 times higher than this setting for all instance sets.

7. Conclusions

This paper has addressed the single machine scheduling problem with distinct time windows and sequence-dependent setup times (SMSPETP). The objective is to minimize the total weighted earliness and tardiness. This problem involves determining the job execution sequence and the starting times of all jobs in the sequence, considering the possibility of inserting idle time between the executions of consecutive jobs. This study also considered the special case with a sequence-independent machine setup time, which is denoted by SMSPETP-SIS.

The main contribution of this study is to propose a new $O(n^2)$ algorithm, named ITIA, for determining the optimal times for the completion of each job in a given sequence. In addition, the development of an implicit enumeration algorithm (IE) and a general variable neighborhood search algorithm (GVNS) also constitute important contributions. The IE algorithm is an exact algorithm, which is only valid for solving SMSPETP-SIS based on theoretical results developed for this problem. Because no specific exact algorithm for solving SMSPETP-SIS is present in the literature, the results obtained by EI can be used as a benchmark in future works. On the other hand, GVNS is a heuristic algorithm that can be applied to solve larger SMSPETP instances. Each sequence generated by the GVNS and IE algorithms is evaluated using ITIA.

Computational experiments performed on a set of instances generated according to the standard parameters from the literature indicated that ITIA is more efficient than the OTA algorithm from Wan and Yen [31]. This is because of the lower times required by IE and GVNS using ITIA rather than OTA.

The IE algorithm is able to solve SMSPETP-SIS in a much lower time compared with CPLEX applied to the formulation of mixed integer programming from Gomes Júnior et al. [13]. Conversely, the GVNS algorithm is robust, finding the optimum solution for all instances with known optimum solutions and zero variability in the remaining cases.

Owing to the better performance achieved by ITIA compared with OTA in the first set of tests, GVNS combined with ITIA (GVNS/ITIA) was compared with the best version of the AGA algorithm from Ribeiro et al. [28], in SMSPETP instances with up to 75 jobs. Two hypothesis tests, a parametric test and a permutation test, demonstrate that GVNS/ITIA is able to determine better average solutions than AGA in a significant number of cases. For example, in instances with 75 jobs, GVNS/ITIA performs better in almost 70% of the instances. These hypothesis tests indicate that with the parameter values set to $5n$, where n is the number of jobs to be scheduled, the GVNS/ITIA algorithm performs better than AGA in relation to the relative average deviation metric. In addition, the average time required by AGA is always 10 times greater than the corresponding time required by GVNS/ITIA.

Regarding future research, the study of new optimality conditions for solutions of SMSPETP-SIS is recommended. In this case, it is expected that the computational time required for the IE algorithm would be reduced, with the consequence of allowing the resolution of larger instances. Furthermore, it is proposed to study optimality conditions for solutions of SMSPETP, in order to apply the IE algorithm in problems with sequence-independent setup times.

Acknowledgements

The authors would like to thank the Minas Gerais State Research Foundation (FAPEMIG), the National Council of Technological and Scientific Development (CNPq), the Minas Gerais Federal Center of Technological Education (CEFET-MG), and the Federal University of Ouro Preto (UFOP) for supporting the development of the present study.

References

[1] Allahverdi A. The third comprehensive survey on scheduling problems with setup times/costs. Eur J Oper Res 2015;246(2):345–78. doi:10.1016/j.ejor.2015.04.004.
[2] Allahverdi A, Gupta JND, Aldowaisan T. A review of scheduling research involving setup considerations. Omega 1999;27(2):219–39. doi:10.1016/S0305-0483(98)00042-5.
[3] Allahverdi A, Ng C, Cheng T, Kovalyov MY. A survey of scheduling problems with setup times or costs. Eur J Oper Res 2008;187(3):985–1032. doi:10.1016/j.ejor.2006.06.060.
[4] Baker KR, Scudder GD. Sequencing with earliness and tardiness penalties: a review. Oper Res 1990;38(1):22–36. doi:10.1287/opre.38.1.22. http://dx.doi.org/10.1287/opre.38.1.22.
[5] Bertsimas D, Tsitsiklis JN. Introduction to Linear Optimization. 1st. Athena Scientific; 1997. ISBN 1886529191.
[6] Carrano E, Wanner E, Takahashi R. A multicriteria statistical based comparison methodology for evaluating evolutionary algorithms. IEEE Trans Evol Comput 2011;15(6):848–70. doi:10.1109/TEVC.2010.2069567.
[7] Davis JS, Kanet JJ. Single-machine scheduling with early and tardy completion costs. Nav Res Logist (NRL) 1993;40(1):85–101. doi:10.1002/1520-6750(199302)40:1⟨85::AID-NAV3220400106⟩3.0.CO;2-C.
[8] Feo TA, Resende MGC. Greedy randomized adaptive search procedures. J Global Optim 1995;6(2):109–33. doi:10.1007/BF01096763.
[9] França Filho MF. Grasp and tabu search applied to scheduling problems in parallel machines (in Portuguese). Phd thesis. School of Electrical and Computer Engineering, State University of Campinas – UNICAMP; Campinas, Brazil; 2007.
[10] Fry TD, Armstrong RD, Blackstone JH. Minimizing weighted absolute deviation in single machine scheduling. IIE Trans 1987;19(4):445–50. doi:10.1080/07408178708975418.
[11] Garey MR, Tarjan RE, Wilfong GT. One-processor scheduling with symmetric earliness and tardiness penalties. Math Oper Res 1988;13(2):330–48. doi:10.1287/moor.13.2.330.
[12] Glover F, Laguna M. Tabu search. Norwell, MA, USA: Kluwer Academic Publishers; 1997. ISBN 079239965X.
[13] Gomes Júnior AC, Carvalho CRV, Munhoz PLA, Souza MJF. A hybrid heuristic method for solving the single machine scheduling problem with earliness and tardiness penalties (in Portuguese). In: Proceedings of the XXXIX Brazilian symposium of operational research – XXXIX SBPO. Fortaleza, Brazil; 2007. p. 1649–60.
[14] Gupta SR, Smith JS. Algorithms for single machine total tardiness scheduling with sequence dependent setups. Eur J Oper Res 2006;175(2):722–39. doi:10.1016/j.ejor.2005.05.018.
[15] Hansen P, Mladenović N, Prez JAM. Variable neighborhood search: methods and applications. 4OR: Q J Oper Res 2008;6(4). 319–316. doi: 10.1007/s10288-008-0089-1.
[16] Janiak A, Janiak WA, Krysiak T, Kwiatkowski T. A survey on scheduling problems with due windows. Eur J Oper Res 2015;242(2):347–57. doi:10.1016/j.ejor.2014.09.043.
[17] Józefowska J. Just-in-time scheduling: models and algorithms for computer and manufacturing systems. International series sn operations research & management science. New York: Springer; 2007. ISBN 9780387717173, http://isbnplus.org/9780387717173.
[18] Kanet JJ, Sridharan V. Scheduling with inserted idle time: problem taxonomy and literature review. Oper Res 2000;48(1):99–110. doi:10.1287/opre.48.1.99.12447.
[19] Kopanos GM, Laínez JM, Puigjaner L. An efficient mixed-integer linear programming scheduling framework for addressing sequence-dependent setup issues in batch plants. Ind Eng Chem Res 2009;48(13):6346–57. doi:10.1021/ie801127t.
[20] Koulamas C. Single-machine scheduling with time windows and earliness/tardiness penalties. Eur J Oper Res 1996;91(1):190–202. doi:10.1016/0377-2217(95)00116-6.
[21] Lee CY, Choi JY. A genetic algorithm for job sequencing problems with distinct due dates and general early-tardy penalty weights. Comput Oper Res 1995;22(8):857–69. doi:10.1016/0305-0548(94)00073-H.
[22] Lourenço HR, Martin OC, Stützle T. Iterated local search. In: Glover F, Kochenberger GA, editors. Handbook of metaheuristics. International Series in Operations Research & Management Science, 57. Springer US; 2003. p. 320–53. doi:10.1007/0-306-48056-5_11.
[23] Mladenović N, Hansen P. Variable neighborhood search. Comput Oper Res 1997;24:1097–100. doi:10.1016/S0305-0548(97)00031-2.
[24] Montgomery DC, Runger GC. Applied statistics and probability for engineers. 6. Wiley; 2013.
[25] Penna PHV, Souza MJF, Gonalves FACA, Ochi LS. A hybrid heuristic algorithm for job scheduling problem on a single-machine (in Portuguese). Production 2012;22(4):766–77. doi:10.1590/S0103-65132012005000020.
[26] Pinedo ML. Scheduling: theory, algorithms, and systems. 4. Springer New York; 2012. doi:10.1007/978-1-4614-2361-4.

[27] Rabadi G, Mollaghasemi M, Anagnostopoulos GC. A branch-and-bound algorithm for the early/tardy machine scheduling problem with a common due-date and sequence-dependent setup time. Comput Oper Res 2004;31(10):1727–51. doi:10.1016/S0305-0548(03)00117-5.

[28] Ribeiro FF, Souza MJF, de Souza SR. An adaptive genetic algorithm to the single machine scheduling problem with earliness and tardiness penalties. In: da Rocha Costa AC, Vicari RM, Tonidandel F, editors. Advances in artificial intelligence - SBIA 2010. Lecture Notes in Computer Science, 6404. Springer Berlin Heidelberg; 2010. p. 203–12. doi:10.1007/978-3-642-16138-4_21.

[29] Souza MJF, Ochi LS, Maculan Filho N. Minimizing earliness and tardiness penalties on a single machine scheduling problem with distinct due windows and sequence-dependent setup times. In: Proceedings of the ALIO/EURO 2008 conference, 1. Buenos Aires: Facultad de Ciencias Exactas y Naturales; 2008. p. 1–6.

[30] Szwarc W, Mukhopadhyay SK. Optimal timing schedules in earliness-tardiness single machine sequencing. Nav Res Logist (NRL) 1995;42(7):1109–14.

[31] Wan G, Yen BP-C. Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties. Eur J Oper Res 2002;142(2):271–81. doi:10.1016/S0377-2217(01)00302-2.

[32] Yano CA, Kim Y-D. Algorithms for a class of single-machine weighted tardiness and earliness problems. Eur J Oper Res 1991;52(2):167–78. doi:10.1016/0377-2217(91)90078-A.