Permutation flowshop problems minimizing core waiting time and core idle time

Permutation flowshop problems minimizing Core Waiting Time and Core Idle Time

Abstract

Waiting time and idle time are among the main cost sources in production systems. They can also affect the feasibility of operations from a technological perspective; hence, both such times have to be kept as small as possible. This paper studies four single-objective variants of the permutation flowshop scheduling problem, where two objectives are considered: the weighted sum of the makespan and the core waiting time, and the weighted sum of the makespan and the core idle time. For each objective, both the problem with the assumption of semi-active solution and the one without it are considered.

A general solution framework for tackling the above-mentioned problems is provided. First, two Mixed Integer Linear Programming (MILP) formulations (based on positional and precedence variables, respectively) and one Constraint Programming (CP) formulation are presented. Second, a MILP-based local search approach based on the positional MILP formulation and the concept of *sliding windows* are defined. An extensive set of computational experiments on benchmark instances show that the positional MILP formulation strongly outperforms the other two formulations in all the considered cases. The experiments also show that the sliding window local search heuristic achieves much better performances than other state-of-the-art local search heuristics. Indeed, it is able to improve the state-of-the-art in 2384

instances out of 2400.

Keywords: Core Waiting Time, Core Idle Time, Permutation Flow Shop, Matheuristics, Scheduling, MILP

---

1. Introduction

Waiting time and idle time are among the main cost sources in production systems. The waiting time is related to the work in process (WIP); in fact, as Little's law tells (Little, 1961), the longer the waiting time, the higher the WIP level, given a fixed required/desired throughput rate. In turn, high WIP levels imply high inventory costs related both to the material and labor content already included in the WIP and to the cost of the space to keep the waiting units. Also, the longer the waiting time, the lower the service level to customers, and hence the lower the competitiveness of the company. A long waiting time is usually related to high utilization levels: as the resource utilization increases, the waiting time (and hence the WIP) increases more than linearly.

The idle time is instead related to a low utilization rate, usually due to resource over-sizing: too fast machines have been acquired or too many workers have been hired with respect to the demand to be satisfied. In this case, the waiting time, and then the WIP, is so small to be immaterial. However, also this situation generates cost: the over-sized resources have been (or have to be) paid, and their cost will be shared among less units with respect to the ones the system would be able to do, thus increasing the unit cost and then decreasing competitiveness.

Beside cost reasons, waiting and idle times should be kept as small as

possible for technological reasons. For example, in many process industries (e.g., chemical or food industries), a too long waiting time after an operation might spoil the WIP and make it unusable for next operations. Also, in those manufacturing systems where operations have to be made on pre-warmed materials, no waiting time can be allowed between warming and manufacturing phases. About idle time, instead, there are cases where resources cannot stay idle between an operation and the next one. This happens, for instance, when machines use consumable, such as paints or powders, that become unusable if the machine stays idle for too long (e.g. paint becomes dried or powder oxidizes).

The impact of waiting and idle times is not the same for any system, as it is also related to the layout of the production system: in single-stage systems, they only depend on the system capacity with respect to the expect demand to be satisfied; in multi-stage systems, the relationship among the capacity of the several stages is also important. In such a case, the impact of waiting and idle times is addressed, at an aggregate level, in the design phase, trying to reach a balanced system.

Once a system has been designed (by considering capacity and demand at an aggregate level), attention should be paid to the short term planning. As processing activities on different jobs can have different times, their schedule on each resource influences the waiting time of the other jobs and the idle time of the resources. For this reason, addressing the scheduling problem by considering also waiting and idle times can help reducing or, at least, keeping under control these two elements.

In multi-stage systems, waiting and idle times are composed of three main

components: front, core and back (waiting and idle) times. The *front waiting time* is the time a job waits before its first operations, while the *back waiting time* is the time a jobs waits, after being completed, that all the other jobs are finished. The *core waiting time*, instead, is the waiting jobs may experiment between one operation and the other. Idle times have the same meaning considering machines: *front idle time* is the time a machine waits before starting the first operation (while other machines are still working), *back idle time* is the time between the moment a machine completes its last operation and the moment the last machine finishes working, and *core idle time* is the time a machine is idle waiting for the next job to be processed. From a cost perspective, all these components are equally relevant; however, from a technological point of view, i.e., when machines have to keep working and/or jobs cannot wait between one operation and the other for technological constraints, core waiting/idle times are the one to reduce, avoid or keep under control.

This paper addresses four variants of the permutation flow shop problem. Given a certain set of jobs J, a permutation flow shop scheduling problem strives for processing the jobs on each machine of a certain set M in the same order, such that a certain objective is minimized. We consider two variants of this problem where the objective functions are: the weighted sum of makespan and total core waiting time, henceforth denoted with CWT_w, and the weighted sum of makespan and total core idle time, henceforth denoted with CIT_w. For both objective functions, two alternative cases are studied: the case where only semi-active schedules are allowed, and the case where such constraint is not considered. A schedule is semi-active

whenever there is no job that can be anticipated without changing the processing order in any of the machines (see Section 3 for more details). The resulting scheduling problems can be indicated by means of the three field notation as $Fm|prmu|CWT_w$, $Fm|prmu, semi\text{-}act|CWT_w$, $Fm|prmu|CIT_w$ and $Fm|prmu, semi\text{-}act|CIT_w$, where the semi-act attribute means that only semi-active schedules are allowed.

The remainder of the paper is the following. The literature is reviewed in Section 2, and some formal definitions are given in Section 3. Section 4 shows the mathematical formulations of the addressed problems, while the proposed solution procedures are described in Section 5, Numerical results are analysed in Section 6 and Section 7 concludes the paper.

2. Literature review

This work addresses the permutation flow shop scheduling problem, a well-studied scheduling problem whose first study, due to Nawaz et al. (1983), dates back to 1983. The most studied objective is the makespan (De Fátima Morais et al., 2022), but a variety of other performance measures, such as the total flow time (Mao et al., 2022), the total tardiness (Framinan and Leisten, 2008; Saber and Ranjbar, 2022), the weighted quadratic tardiness (Silva et al., 2022), the energy consumption (Öztop et al., 2020), etc., have been considered as well. In this paper, the core idle and waiting times are considered as performance measures to be optimized. In the following, the literature about these two performance measures is reviewed separately.

2.1. Waiting time minimization

The waiting time minimization has been studied in various system layouts. The so called no-wait constraint, imposing that the no waiting time is allowed among operations, has been included in the formulation of different scheduling problems (Allahverdi et al., 2020; Sharma et al., 2020). The reader can refer to Allahverdi (2016) for a survey about this topic. In the special case of single-machine systems, total and front waiting times are equal, and schedules are semi-active in all the cases. Bagchi (1989) addressed the problem of minimizing the mean front waiting time and the total absolute difference of front waiting times. Some authors addressed the problem of minimizing the total waiting time in single-machine systems with specific characteristics, such as setup times (sequence-dependent in Soroush (2012), and sequence-independent in de Matta (2019)). For the single-machine environment, the minimization of the variance of waiting times has been largely studied in the literature, and it was proved to be a NP-hard problem (Kubiak, 1993). Eilon and Chowdhury (1977) proved that the optimal schedule for this problem has a V-shaped distribution of processing times. Some authors discussed the relationships between the variance of waiting times and of completion times (Merten and Muller, 1972; Zhou and Cai, 1996), and found some analytical properties and other influencing factors (Li et al., 2007). Heuristic algorithms to solve this problem were proposed in Ye et al. (2007), while Xu (2011) solved the weighted version of the problem to optimality. Similarly, the weighted sum of squared waiting times was addressed by Szwarc and Mukhopadhyay (1995); Sun et al. (2011), the former by developing a decomposition scheme to solve the problem, and the latter by focusing on the case of deteriorating

jobs. In a single-stage system, also the case of parallel machines and waiting time variance minimization has been addressed (Xu and Ye, 2007).

The waiting time minimization has also been studied in more complex layouts, such as job shops (Chu and Portmann, 1993) and flow shops (Maassen et al., 2019). The total waiting time was minimized in Chu and Portmann (1993) through a heuristic algorithm to find efficient semi-active schedules for the job shop scheduling problem. In permutation flow shops, Maassen et al. (2019, 2020) addressed instead only the minimization of core waiting time in a permutation flow shop with semi-active schedules. Specifically, Maassen et al. (2019) proposed a NEH-based heuristic to solve the problem, while Maassen et al. (2020) focused on the relationship between core waiting time ad total completion time. Also, Birgin et al. (2020) addressed the permutation flow shop with semi-active schedules, and they proposed a beam search meta-heuristic to first minimize the earliness and tardiness and, then, among the optimal solutions, to find the schedule that minimizes the core waiting time. Finally, core waiting time was minimized in a permutation flow shop with general schedules by De Abreu and Fuchigami (2022), which proposed a comparison of four Mixed Integer Liner Programming (MILP) models to minimize the sum of core waiting time and front and core idle times.

The problem of minimizing the waiting time has also been addressed in contexts related to service systems, such as the minimization of passenger waiting times in transportation by exploiting Markov chains (Lees-Miller, 2016), or the minimization of patient waiting time in hospitals through the use of simulations (van Essen et al., 2012).

2.2. Idle time minimization

As discussed in Section 1, minimizing the machine idle time is very relevant in production processes, because of its tight relationship with the machine utilization. The constraint of having zero idle time (*no-idle*) is common in production scheduling problems (Bektaş et al., 2020; Goncharov and Sevastyanov, 2009; Maassen et al., 2020). The possibility of limiting the number of machine interruptions, which is strictly related to limiting the idle time (Höhn et al., 2012), has also been addressed.

Instead of imposing constraints on the idle time, some authors considered the problem of minimizing it, or some of its components, in various system layouts. Some meta-heuristics were proposed by Liao et al. (2007) and Yagmahan and Yenisey (2008) to minimize the total idle time in flow shops. Specifically, Liao et al. (2007) developed a particle swarm optimization algorithm, while Yagmahan and Yenisey (2008) an ant colony meta-heuristic to minimize together makespan, flow time, and total machine idle time. Other meta-heuristics were proposed by Hosseini and Tavakkoli-Moghaddam (2013) for a flow shop system with two stages, to minimize together the total idle time and the mean deviation from a common due data. Also, as already mentioned, De Abreu and Fuchigami (2022) considered the minimization of the sum of front and core idle time (and of the core waiting time) in the permutation flow shop scheduling problem with general schedules.

The minimization of the core idle time has only been addressed by Liu et al. (2016), which considered a permutation flow shop system. A NEH-based heuristic is proposed to minimize the weighted sum of makespan and core idle time, with the assumption of semi-active schedules.

This paper addresses the minimization of core waiting and idle times in a permutation flow shop with both general and semi-active schedules. Similar problems were addressed in De Abreu and Fuchigami (2022); Liu et al. (2016); Maassen et al. (2019). While De Abreu and Fuchigami (2022) considered general schedules, Liu et al. (2016); Maassen et al. (2019) assumed semi-active schedules. Also, De Abreu and Fuchigami (2022) addressed the minimization of the sum of waiting and idle times, Liu et al. (2016) solved the problem of minimizing the weighted sum of makespan and core idle time, while Maassen et al. (2019) focused on the minimization of the core waiting time. Lastly, De Abreu and Fuchigami (2022) proposed four MILP formulations and several multi-warm procedures to solve them, while Liu et al. (2016); Maassen et al. (2019) both proposed NEH-based heuristics as solution procedures. This paper differs from the state-of-the-art as it proposes a unique and general solution framework to consider alternatively the minimization of the weighted sum of core waiting time and makespan, and the minimization of the weighted sum of core idle time and makespan. The framework considers both assumptions on semi-active and general schedules. Two MILPs and one Constraint Programming (CP) model are proposed, and the solution procedure involves the use of local searches that exploit the MILP models. Such framework allows to reach better solutions with respect to the state-of-the-art approaches.

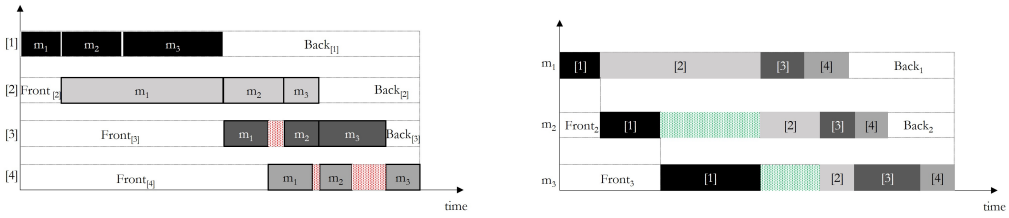
3. Formal definitions

In the following, first the concepts of idle and waiting times are formally introduced, together with the formal definition of the problems addressed in

the paper; second, the difference with regard to optimal solutions between semi-active and general schedules is given.

3.1. Waiting and idle times

As mentioned in Section 1, idle and waiting times are composed of three main components: front, core and back. These components are here defined, for a flow shop, according to the formal definition proposed in Maassen et al. (2020). Figure 1 shows an illustrative example: the two graphics represent the machine-oriented (Figure 1a) and the job-oriented (Figure 1b) Gantt charts, respectively, for a permutation flow shop composed of three machines (namely, m_1, m_2, m_3) in which four jobs flow (in the Gannt chart, $[k]$ indicates the job in the $k-$th position of the schedule).



(a) Job-oriented Gantt chart (b) Machine-oriented Gantt chart

Figure 1: Illustrative example of waiting and idle time components in Gantt charts (Maassen et al., 2020)

The job-oriented Gantt chart in Figure (1a) graphically shows the three components of the waiting time. The front waiting time ($Front_{[k]}$ in the figure) is the time that the job $[k]$ of a schedule waits before starting its first operation. Instead, the back waiting time ($Back_{[k]}$ in the figure) is the time job $[k]$ with $k < |P|$ waits after its completion and the completion of the last completed job (i.e., job $[|P|]$). The core waiting time $CWT_{k,m}$ is

the time job $[k]$ waits on machine $m \in M - \{1\}$ after having completed its operation on machine $m-1$ (red-dotted areas in Figure (1a)); the total core waiting time is then the sum over all jobs and machines of $CWT_{k,m}$, i.e., $CWT = \sum_{m \in M-\{1\}} \sum_{k \in P} CWT_{k,m}$.

Conversely, the machine-oriented Gantt chart in Figure (1b) graphically shows the three components of the idle time. The front idle time ($Front_m$ in the figure) is the time machine $m \in M - \{1\}$ is idle waiting for job [1], starting from the time machine $m = 1$ starts processing it. The back idle time ($Back_m$ in the figure) is the time machine $m < |M|$ is idle from when it completes the last job till when the last machine (i.e., machine $|M|$) finished its processing. The core idle time $CIT_{m,k}$ is the time machine m is idle waiting for job in position $k \in P - \{1\}$ after having processed job in position $[k-1]$ (i.e., the green-dotted areas in Figure (1b)); the total core idle time is then $CIT = \sum_{m \in M} \sum_{k \in P-\{1\}} CIT_{m,k}$.

3.2. General and semi-active schedules

According to the definition of Pinedo (2012), a *semi-active* schedule is defined as a sequence of jobs in which jobs start as early as possible on each machine, which means that the corresponding Gantt-chart is left-shifted. With the assumption of semi-active schedules, the first machine has no front and core idle time, and the first job has no front and core waiting times. A *general* schedule, instead, is a schedule in which the semi-active assumption is relaxed. Thus, the first machine can be idle, and the first job can wait before being processed on some machines.

In the case of regular objective functions, such as minimizing the makespan or the total completion time, the optimal solution does not change if the semi-

active assumption is included in the problem definition. However, if other objective functions such as core waiting and core idle times are involved, the same no longer holds. The following proposition proves it.

Proposition 3.1. In a permutation flow shop scheduling problem with a set of jobs J and set of machines M, and core waiting time or core idle time as objective function, the optimal solution under the semi-active schedule assumption might be non-optimal with respect to the general assumption.

Proof. The proof is made by two counter-examples, one for the core waiting time and the other for the core idle time. Both examples consider a permutation flow shop with two jobs $\{j_1, j_2\}$ and three machines $\{m_1, m_2, m_3\}$. The possible job sequences on each machine are: $S_1 = (j_1, j_2)$ and $S_2 = (j_2, j_1)$. Both sequences are evaluated, for each example, with the semi-active assumption (SA) and without it (Gen). Table 1 shows, for each schedule, the makespan (C_{max}), the total completion time (C_{tot}), the total core waiting time (CWT), and the total core idle time (CIT). The Gantt charts of the four schedules are displayed in Figures 2 and 3 for the examples on waiting time and idle time, respectively.

The first example is related to the waiting time. The processing times of jobs on each machine are as follows: job j_1 has processing times equal to $\{19, 54, 5\}$, while job j_2 has processing times equal to $\{19, 22, 77\}$ on machines $\{m_1, m_2, m_3\}$, respectively. From Table 1 and from the Gantt charts in Figure 2, it is possible to see that, if the semi-active assumption is made and CWT is taken as objective function, then the optimal schedule is S_2, with an optimal CWT value equal to 26. However, if general schedules are allowed, then the optimum is $CWT = 0$, obtained by both sequences. The waiting time is

Table 1: Numerical examples: summary of performance measures.

ID Example	Sequence	Schedule type	C_{max}	C_{tot}	CWT	CIT
1	(j_1, j_2)	SA	172	250	35	
1	(j_1, j_2)	Gen	172	250	0	
1	(j_2, j_1)	SA	123	241	26	
1	(j_2, j_1)	Gen	123	241	0	
2	(j_1, j_2)	SA	74	142		3
2	(j_1, j_2)	Gen	74	146		0
2	(j_2, j_1)	SA	97	137		52
2	(j_2, j_1)	Gen	97	137		0

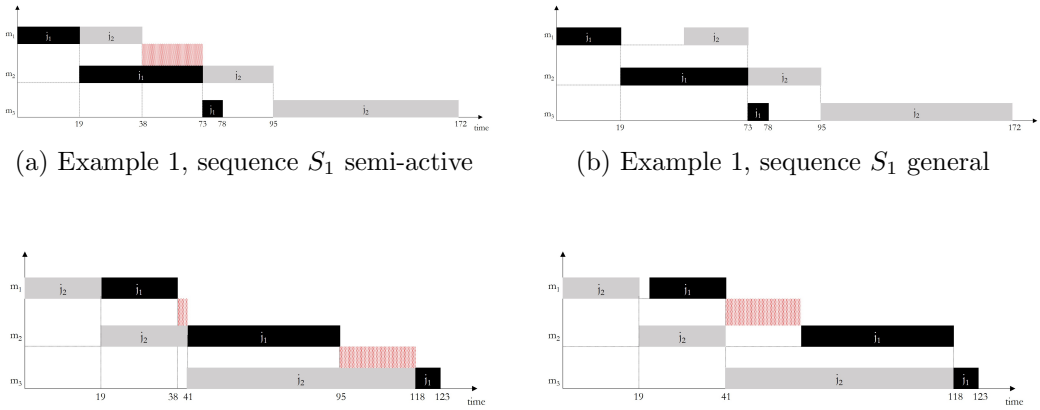
shown in Figure 2 with the red-dotted areas.

In the second example (bottom part of Table 1, Figure 3), the idle time is considered. In this case, the processing times of jobs on machines are $\{9, 54, 5\}$ for job j_1 and $\{29, 9, 2\}$ for j_2 on machines $\{m_1, m_2, m_3\}$, respectively. In this case, if core idle time is considered, under the semi-active assumption the optimum is $CIT = 3$, provided by S_1, while, if general schedules are allowed, then the optimum is $CIT = 0$, reached by both sequences. The idle time is shown in Figure 3 with the green-dotted areas.

□

4. Mathematical formulations

This section presents three different formulations for each of the problems defined in Section 3. The first two formulations are MILP, which rely on positional variables and precedence variables, respectively. The last formulation is based on Constraint Programming and makes use of global constraints and structures included in the library of IBM ILOG CP Optimizer (Laborie



(a) Example 1, sequence S_1 semi-active

(b) Example 1, sequence S_1 general

(c) Example 1, sequence S_2 semi-active

(d) Example 1, sequence S_2 general

Figure 2: Proof of Proposition 3.1. Example 1, Gantt chart of the four schedules.

et al., 2018).

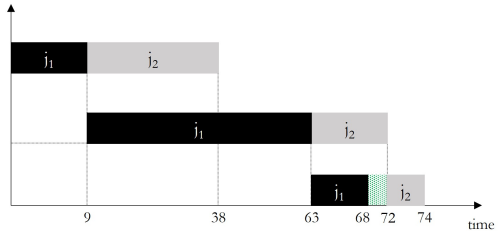
In the models, M is the set of machine indexes $\{1, \ldots, |M|\}$, while J is the set of job indexes $\{1, \ldots, |J|\}$. As in Section 3, P is used to refer to the set of all the possible positions $\{1, \ldots, |J|\}$ where a job can be scheduled. The considered parameters are:

- $p_{j,m} \in \mathbb{R}_+$ is the processing time of job $j \in J$ on machine $m \in M$;
- $\alpha \in [0, 1]$ is the weight of C_{max} in the objective function;
- $K \in \mathbb{R}_+$ is a large enough constant.

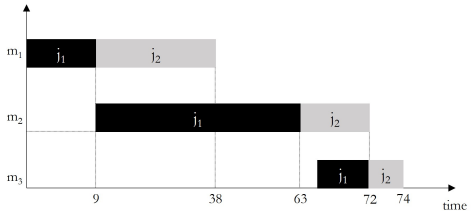
For all models, the objective functions $o.f.$ can be:

$$CIT_w = wC_{max} + (1 - w)CIT \qquad CWT_w = wC_{max} + (1 - w)CWT \quad (1)$$

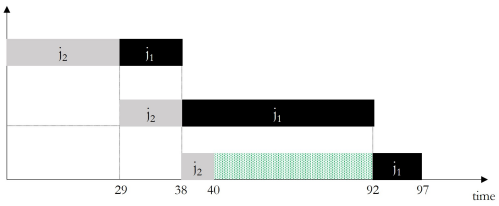
where the calculations of C_{max}, CIT, CWT will be computed differently for each model.



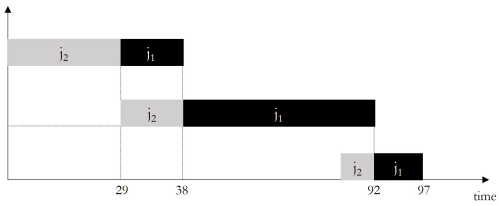
(a) Example 2, sequence S_1 semi-active



(b) Example 2, sequence S_1 general



(c) Example 2, sequence S_2 semi-active



(d) Example 2, sequence S_2 general

Figure 3: Proof of Proposition 3.1. Example 2, Gantt chart of the four schedules.

4.1. MILP with positional variables

The MILP formulation based on positional variables, and denoted by **`MILP-POS`**, is presented in this section.

The considered decision variables are:

- $x_{j,k} \in \{0, 1\}$: binary variable equal to 1 if job $j \in J$ is scheduled at position $k \in P$, 0 otherwise;
- $c_{k,m} \in \mathbb{R}_+$: completion time of the job at position $k \in P$ on machine $m \in M$;

In the case where the schedules may not be semi-active, the following formulation holds:

$$
\begin{aligned}
\min \quad & o.f. && && (2)\\
\text{s.t.} \quad & \sum_{j\in J} x_{j,k} = 1 && \forall k \in P && (3)\\
& \sum_{k\in P} x_{j,k} = 1 && \forall j \in J && (4)\\
& c_{1,1} = \sum_{j\in J} x_{j,1} p_{j,1} && && (5)\\
& c_{k,m} \geq c_{k,m-1} + \sum_{j\in J} x_{j,k} p_{j,m} && \forall k \in P, m \in M - \{1\} && (6)\\
& c_{k,m} \geq c_{k-1,m} + \sum_{j\in J} x_{j,k} p_{j,m} && \forall k \in P - \{1\}, m \in M && (7)\\
& x_{j,k} \in \{0,1\} && \forall j \in J, k \in P && (8)\\
& c_{k,m} \in \mathbb{R}_+ && \forall m \in M, k \in P && (9)
\end{aligned}
$$

The components of $o.f.$ are:

$$
\begin{aligned}
C_{max} &= c_{|J|,|M|}\\
CIT &= \sum_{m\in M} \left(c_{|J|,m} - c_{1,m} - \sum_{k\in P-\{1\}} \sum_{j\in J} p_{j,m} x_{j,k} \right)\\
CWT &= \sum_{k\in P} \left(c_{k,|M|} - c_{k,1} - \sum_{m\in M-\{1\}} \sum_{j\in J} p_{j,m} x_{j,k} \right)
\end{aligned}
\qquad (10)
$$

Constraints (3) state that each position is assigned to a single job, while constraints (4) impose that each job is assigned to a single position. Constraints

(5) force the starting time of the first job on the first machine to be equal to zero. Constraints (6) are the flow shop constraints establishing that a job can start to be processed on machine $m \in M - \{1\}$ once completed by the previous machine. Constraints (7) impose that the job at position $k \in P - \{1\}$ starts being processed by a machine $m \in M$ after the completion time of the job at position $k-1 \in P$. Finally, (8) - (9) define the domain of the decision variables.

In order to impose that the schedules are semi-active, the following constraints must be added to the model:

$$c_{k,1} = c_{k-1,1} + \sum_{j \in J} x_{j,k} p_{j,1} \quad \forall k \in P - \{1\} \tag{11}$$

$$c_{1,m} = c_{1,m-1} + \sum_{j \in J} x_{j,1} p_{j,m} \quad \forall m \in M - \{1\} \tag{12}$$

Constraints (11) impose that there is no idle time on the first machine. Constraints (12) force the waiting time related to the first job to be zero.

4.2. A MILP Formulation Based on Precedence Variables

This formulation is referred to as **`MILP-PREC`** in the rest of the paper.

The considered decision variables are:

- $z_{i,j} \in \{0, 1\}$: binary variable equal to 1 if job $i \in J$ is scheduled before another job $j \in J$, 0 otherwise;
- $c_{j,m} \in \mathbb{R}_+$: completion time of job $j \in J$ on machine $m \in M$;
- $b_{m,min} \in \mathbb{R}_+$ is the minimum starting time of a job on machine $m \in M$;

- $c_{m,max} \in \mathbb{R}_+$ is the maximum completion time of a job on machine $m \in M$;
- $c_{m,min} \in \mathbb{R}_+$: the minimum completion time of a job on machine $m \in M$;

The formulation, which allows general schedules, is as follows:

$$\begin{aligned}
\min \quad & o.f. && && (13)\\
\text{s.t.} \quad & z_{i,j} + z_{j,i} = 1 && \forall i,j \in J : i \neq j && (14)\\
& z_{i,j} + z_{j,k} + z_{k,i} \leq 2 && i,j,k \in J : i \neq j \neq k && (15)\\
& c_{j,m} \geq c_{j,m-1} + p_{j,m} && \forall j \in J, m \in M - \{1\} && (16)\\
& c_{j,m} - p_{j,m} \geq c_{i,m} - K z_{j,i} && \forall i,j \in J : i \neq j, m \in M && (17)\\
& c_{j,m} \leq c_{m,max} && \forall j \in J, m \in M && (18)\\
& c_{j,m} - p_{j,m} \geq b_{m,min} && \forall j \in J, m \in M && (19)\\
& z_{i,j} \in \{0,1\} && \forall i,j \in J : i \neq j && (20)\\
& c_{j,m} \in \mathbb{R}_+ && \forall j \in J, \forall m \in M && (21)\\
& c_{m,max} \in \mathbb{R}_+ && \forall m \in M && (22)\\
& b_{m,min} \in \mathbb{R}_+ && \forall m \in M && (23)
\end{aligned}$$

The components of $o.f.$ are:

$$C_{max} = c_{|M|,max}$$

$$CIT = \sum_{m \in M} (c_{m,max} - b_{m,min} - \sum_{j \in J} p_{j,m})$$

$$CWT = \sum_{j \in J} (c_{j,|M|} - c_{j,1} - \sum_{m \in M - \{1\}} p_{j,m})$$

It can be noticed that variables $c_{m,max}$ and $b_{m,min}$ are not necessary in case CWT is used as objective function.

Constraints (14) ensure that $i \in J$ precedes $j \in J$ or vice-versa. Constraints (15) guarantee that there is no precedence loop. Constraints (16) and (17) impose the flow-shop constraints and the precedence constraints over each machine. Constraints (18) - (19) ensure that $b_{m,min}$ and $c_{m,max}$ are correctly related with the other variables when minimizing the objective function. Finally, (20)-(23) state the domain of the decision variables.

In the case semi-active schedules are assumed, the following variables and constraints need to be included in the model:

$$c_{|M|,min} \in \mathbb{R}_+ \tag{24}$$

$$c_{1,max} = \sum_{j \in J} p_{j,1} \tag{25}$$

$$c_{j,|M|} \leq c_{|M|,min} + \sum_{i \in J} K x_{i,j} \ \forall j \in J \tag{26}$$

$$c_{|M|,min} \leq (1 + \sum_{i \in J} K x_{i,j}) \sum_{m \in M} p_{j,m} \ \forall j \in J \tag{27}$$

A new variable $c_{|M|,min} \in \mathbb{R}_+$ is introduced in (24). Constraint (25) enforces that there is no idle time on the first machine. Constraint (26) and (27)

impose that:

- $c_{|M|,min}$ is equal to the minimum completion time of a job in the last machine, when minimizing the objective function;
- there is no waiting time associated with the first job of the sequence [1], hence $c_{[1],|M|} = \sum_{m \in M} p_{[1],m}$.

4.3. A Constraint Programming Formulation

The CP formulation presented in this section, based on interval variables, is indicated with **CP-IV** from this point forward.

The considered decision variables are:

- $T_{j,m}$: interval variable related to the j-th job scheduled at the m-th machine;
- S_m: sequence variable including all the jobs processed at machine $m \in M$;
- C_{max}: integer variable used to indicate the makespan.

The following CP formulation holds:

$$\min \quad o.f. \tag{28}$$

$$\text{s.t.} \quad S_m = SequenceVar(\{T_{j,m}\}_{j \in J}) \qquad \forall m \in M \tag{29}$$

$$sameSequence(S_1, S_m) \qquad \forall m \in M - \{1\} \tag{30}$$

$$noOverlap(S_m) \qquad \forall m \in M \tag{31}$$

$$endBeforeStart(T_{j,m}, T_{j,m+1}) \qquad \forall J \in J, m \in M - \{|M|\} \tag{32}$$

$$C_{max} \geq endOf(T_{j,|M|}) \qquad \forall j \in J \tag{33}$$

The components of $o.f.$, which has to be minimized (28), can be expressed as follows:

$$CIT = \sum_{m \in M} (\max_{j \in J}\{endOf(T_{j,m})\} - \min_{j \in J}\{startOf(T_{j,m})\} - \sum_{j \in J} p_{j,m})$$

$$CWT = \sum_{j \in J} (endOf(T_{j,|M|}) - startOf(T_{j,1}) - \sum_{m \in M} p_{j,m})$$

Constraints (29) and (30) ensure that the jobs are performed in the same order in all the machines. Constraints (31) enforce that the jobs assigned to each machine do not overlap in time. Constraints (32) are flowshop constraints, while Constraints (33) state that C_{max} is greater than the completion times of the jobs on the last machine.

In the case semi-active schedules are assumed, the following constraints must be included in the model:

$$\max_{j \in J}(endOf(T_{j,1})) = \sum_{j \in J} p_{j,1} \tag{34}$$

$$If \min_{j \in J}(endOf(T_{j,1})) = endOf(T_{[1],1})) \; then \; endOf(T_{[1],|M|}) = \sum_{m \in M} p_{[1],m} \; \forall j \in J \tag{35}$$

Constraint (34) imposes that there is no idle time on the first machine, while (35) forces the first job of the sequence to have zero core waiting time.

5. Solution algorithms

Three local search based matheuristics are developed to find solutions of the considered problems. The structure of the algorithms is the same for both the considered objective functions of (1) (CIT_w, CWT_w) and for both semi-active and general schedule assumptions. For all the versions of the problem, the local search algorithms exploit the positional MILP formulation **MILP-POS**. Matheuristics are hybrid solution approaches that couple heuristic frameworks with exact methods. In this case, for example, neighbourhoods are defined as integer programs and their exploration is performed by a MILP solver. The sliding window local search starts from an initial solution and, iteratively, fixes a subset of the solution and the non-fixed part is re-optimized through **MILP-POS**. Instead, the one-opt and swap local searches start from an initial solution, iteratively change part of it with different rules, and use **MILP-POS** to find the optimal timing related to the new solution; in this case **MILP-POS** is reduced to a LP model, as all the binary variables related to the schedule are fixed, while all the continuous variables related

to the timing of the sequence must be optimized.

5.1. Sliding window local search

Let $\mathcal{N}_{\pi,\psi,w_P}$ be the neighbourhood structure of a given solution π, with an initial position ψ of the solution π and a *size* parameter w_P. The *job-window* $\pi(w_P, \psi)$ is the index set of jobs located in the consecutive positions $[\psi], [\psi+1], \ldots, [\psi+w_P-1]$ of the schedule π. The neighbourhood $\mathcal{N}_{\pi,\psi,w_P}$ is composed of all the solutions that can be obtained by fixing all the positions of π not included in $\pi(w_P, \psi)$. The positions in $\pi(w_P, \psi)$ are optimized by means of the **MILP-POS**.

The pseudo-code of the local search is devised in Algorithm 1. The algorithm takes an initial solution π, a window size w_P and a time limit θ as input, and gives as output the heuristic solution π^*. At each iteration (lines 3-13 of Algorithm 1), the best solution in $\mathcal{N}_{\pi,\psi,w_P}$ is found by means of **MILP-POS** and stored if is better than the best solution found so far. The w_P parameter needs to be carefully chosen, since the solution of each sub-problem (line 4 of Algorithm 1) has an exponential time complexity with respect to this parameter. At each iteration, the first job position ψ increases by one unit, thus the job-window *slides*. The algorithm stops when the time limit θ is reached.

5.2. One-opt and swap local searches

The structure of the local searches based on the one-opt and swap neighbourhoods is similar to that using the sliding windows. The one-opt local search works as follows: until the time limit θ is not reached, for each position $[k]$ from $|J|$ - 1 to 1, jobs at positions $[k]$ and $[k-1]$ are swapped and

Algorithm 1 Sliding window local search

Input: π, w_P, θ
Output: π^*

$\psi \leftarrow 0$;
$OF^* \leftarrow MAXVALUE$;
while θ is not reached **do**
    $\pi \leftarrow$ best solution in $\mathcal{N}_{\pi,\psi,w_P}$ computed by means of **MILP-POS**;
    Evaluate o.f. OF_π of π;
    if $OF_\pi < OF^*$ **then**
        $OF^* \leftarrow OF_\pi$;
        $\pi^* \leftarrow \pi$;
    end if
    $\psi \leftarrow \psi + 1$;
    if $\psi > |P| - w_P$; **then**
        $\psi \leftarrow 0$
    end if
end while

the new solution is evaluated by means of **MILP-POS**. When the stopping criterion is reached, the best solution found is the final solution π^* given as output of the algorithm. For the swap local search, the same mechanism is applied; however, instead of swapping positions $[k]$ and $[k-1]$, all the possible swaps between $[k]$ and $[l]$ (with $l > k$) are evaluated. For both algorithms, **MILP-POS** is used only to evaluate the objective function; indeed, the sequence is fixed at each iteration (all the binary scheduling variables are fixed), thus **MILP-POS** reduces to a LP problem to be solved.

5.3. Initial solution

The initial solution π can be generated in many ways (e.g., randomized, constructive heuristics, etc.). In this paper, the NEH-based heuristics of Maassen et al. (2019) and Liu et al. (2016) are used to find the initial solutions of the problems with, respectively, CWT_w and CIT_w objective functions.

Both algorithms are based on the standard NEH constructive algorithm, but develop ad-hoc methods to initially sort jobs. The reader is referred to these papers for further details on the technical aspects of the NEH heuristics. However, the general NEH procedure follows:

1. All jobs are sorted according to a specific rule: for CWT_w problems, jobs are sorted according to the descending order of the index IF_{NEH_M} defined by Maassen et al. (2019); for CIT_w problems, jobs are sorted according to the descending order of the index PR_{LJP} defined by Liu et al. (2016).

2. The first two jobs are scheduled and the sub-sequence with the smallest $o.f.$ (either CIT_w or CIT_w, evaluated by exploiting the MILP formulation) is chosen.

3. The next unscheduled job is assigned to the sub-sequence and scheduled in each possible position to find the schedule with the minimum $o.f.$ value. This step is iterated for all the remaining jobs.

4. For the CIT_w problem, if ties exist, the tie-braking rule proposed in Liu et al. (2016) is used.

Maassen et al. (2019) and Liu et al. (2016) actually solves the problems with semi-active schedules. However, in the proposed local searches, the found schedule is used as initial solution both in the case of semi-active and general schedules. Moreover, while Liu et al. (2016) uses the CIT_w objective function, Maassen et al. (2019) minimizes only the core waiting time. In our

approach, the NEH by Maassen et al. (2019) is slightly modified such that all the partial and complete sequences are evaluated in terms of CWT_w.

6. Computational experiments

The developed algorithms have been tested on the well-known Taillard (Taillard, 1993) and Vallada–Ruiz–Framinan (VRF) (Vallada et al., 2015) benchmarks.

The Taillard benchmark is composed of 12 different problems in terms of number of jobs and machines. The number of jobs varies between 20 and 500, while the number of machines between 5 and 20. For each problem, 10 instances are available, for a total number of 120 instance.

The VRF benchmark is composed of 480 instances, and it includes both small and large problems (for each problem, 10 instances are available). The small problems range between 10 and 60 jobs, and 5 and 20 machines. Large problems, instead, include from 100 to 800 jobs, and from 20 to 60 machines.

Summing up, 600 instances are available in total.

Two types of computational experiments have been carried on. The first experiment deals with the comparison among the mathematical formulations presented in Section 4 on a subset of instances. The aim is to assess the performances of such formulations, both in terms of the quality of the feasible solutions and lower bounds provided within a given time limit.

The second experiment compares the three heuristic algorithms presented in Section 5 to evaluate the efficiency of the proposed approaches. For the problems with semi-active schedules, the proposed local searches are also compared with the state-of-the-art NEH algorithms of Maassen et al. (2019)

and Liu et al. (2016), to assess the improvement with respect to the literature.

CPLEX 12.10 and CP Optimizer 12.10 by IBM ILOG are used as MILP and CP solvers, respectively. All the algorithms are implemented in Java through the IBM ILOG Concert Technology. Tests are run on a Intel(R) Core(TM) i7-8700K CPU processor at 3.70 GHz, with 32 GB of RAM;

For reason of conciseness, in all the experiments the weights of the objective functions CWT_w and CIT_w are set to 0.5. However, preliminary results shown that the same results and considerations can be found for the other weights. Detailed results of all the experiments are available upon requests to the authors.

6.1. Computational assessment of the MILP and CP Models

This section compares the performances of the MILP and CP models on a subset of 180 instances corresponding to the instance in Taillard and VRF benchmarks with up to 100 jobs. For each combination of values of n and m, 5 of the 10 available instances are randomly selected and solved by all the three models with a time limit of 5 minutes. For each instance, the models are tested on the four problems previously defined (o.f.: CWT_w, CIT_w; schedule: general, semi-active). For each instance-problem-model, upper-bound (ub), lower-bound (lb), and computation time (t) have been collected.

The quality of the results provided by the models is evaluated by (i) the Average Percentage Gap (APG) and (ii) the Average Relative Percentage Deviation (ARPD). (i) The percentage gap of the model k on the instance i is computed as $PG_{k,i} = \frac{ub_{k,i} - lb_{k,i}}{ub_{k,i}}$. The average percentage gap of a model k over a set of instances I is the average over all instances $i \in I$ of $PG_{k,i}$. This value represents both the quality of the upper and lower bounds provided by

the given model, and it is equal to zero when all instances in I are solved to optimality. (ii) The relative percentage deviation of k on i is computed as $RPD_{k,i} = 100 \times \frac{ub_{k,i} - ub_{min}}{ub_{min}}$, where $ub_{min} = \min_{k \in K} ub_{k,i}$ (i.e., ub_{min} is the best ub found for the instance i among the available models in $K = \{\texttt{MILP-POS}, \texttt{MILP-PREC}, \texttt{CP-IV}\}$). The average relative percentage deviation of k over a set of instances I is the average over all instances $i \in I$ of $RD_{k,i}$. This value measures the quality of the solutions computed by the model k compared to other available models.

Tables 4 and 5 show the summary of the experimental campaign in terms of APG and ARPD values for all the considered instances, grouped by the number of jobs n. In each table, the last row shows the average APG and ARPD values for each model over all the instances.

The results clearly indicate that the best performing model in terms of APG is the positional MILP model (**MILP-POS**). This finding holds for all the four considered problem variants. The other two models (**MILP-PREC** and **CP-IV**) achieve poor values of APG, since they are not relying on a strong lower bound. More specifically, these models are not capable of computing any lower bound different from 0, for most of the instances of $Fm|prmu|CIT_w$ and $Fm|prmu, \textit{semi-act}|CIT_w$. This is due to the fact that **MILP-PREC** yields a weak continuous relaxation and **CP-IV** relies on a trivial combinatorial bound.

When the ARPD is considered, the trend is slightly different. **MILP-POS** still outperforms the other models for the problems with general schedules, while **CP-IV** yields better performance when semi-active schedules are considered. This indicates that the automatic search strategy used by IBM

	General						Semi-active					
	MILP-POS		MILP-PREC		CP-IV		MILP-POS		MILP-PREC		CP-IV	
n	**APG**	**ARPD**	**APG**	**ARPD**	**APG**	**ARPD**	**APG**	**ARPD**	**APG**	**ARPD**	**APG**	**ARPD**
10	**0.0**	**0.0**	**0.0**	**0.0**	50.9	15.9	**0.0**	**0.0**	**0.0**	**0.0**	25.4	**0.0**
20	**22.0**	**0.2**	49.9	2.5	61.1	28.8	**43.3**	**1.3**	73.0	8.1	75.2	3.1
30	**33.3**	**0.0**	72.2	9.5	73.6	33.5	**62.8**	8.1	88.3	39.4	80.4	**0.1**
40	**38.2**	**0.0**	80.5	12.3	75.8	29.3	**70.8**	15.2	93.5	68.6	83.9	**0.0**
50	**38.6**	**0.0**	85.9	22.0	77.7	28.9	**72.4**	14.4	96.4	104.9	86.9	**0.0**
60	**43.8**	**0.0**	89.2	29.8	79.7	26.2	**77.6**	27.3	97.7	160.2	88.2	**0.0**
100	**60.4**	**0.8**	94.9	84.7	82.7	12.3	**87.3**	54.4	98.9	137.7	93.6	**0.0**
Overall	**33.9**	**0.1**	72.7	22.9	80.0	25.1	**60.9**	16.7	80.0	73.0	77.5	**0.6**

Table 2: APG and ARD results for VRF and Taillard instances for the problem related to CWT_w (%).

	General						Semi-active					
	MILP-POS		MILP-PREC		CP-IV		MILP-POS		MILP-PREC		CP-IV	
n	**APG**	**ARPD**	**APG**	**ARPD**	**APG**	**ARPD**	**APG**	**ARPD**	**APG**	**ARPD**	**APG**	**ARPD**
10	**0.0**	**0.0**	35.4	0.3	100.0	8.0	**0.0**	**0.0**	25.1	**0.0**	100.0	0.5
20	**2.9**	**0.0**	100.0	4.9	100.0	14.4	**14.8**	**0.4**	100.0	11.6	100.0	4.5
30	**6.4**	**0.0**	100.0	11.2	100.0	17.6	**23.2**	**0.5**	100.0	33.4	100.0	4.5
40	**7.5**	**0.0**	100.0	17.0	100.0	16.4	**26.0**	**1.5**	100.0	51.2	100.0	4.0
50	**5.8**	**0.0**	100.0	20.5	100.0	16.9	**21.6**	**0.2**	100.0	61.1	100.0	4.5
60	**7.4**	**0.0**	100.0	24.6	100.0	17.5	**28.1**	**3.0**	100.0	71.9	100.0	**3.0**
100	**29.4**	**0.5**	100.0	30.1	100.0	5.3	**46.0**	29.0	100.0	48.8	100.0	**0.7**
Overall	**8.4**	**0.1**	100.0	15.4	91.4	13.8	**24.8**	4.8	91.4	39.4	100.0	**3.3**

Table 3: APG and ARD results for VRF and Taillard instances for the problem related to CIT_w (%).

ILOG CP Optimizer 12.10 achieves impressive performance, even without any aid from an effective bounding algorithm. This allows to compute high quality upper bounds by means of **CP-IV**. A likely reason for this is that the inclusion of the semi-active constraints allows **CP-IV** to achieve a much more effective propagation.

6.2. Computational assessment of the local search procedures

The VRF and Taillard benchmarks have been used to compare the three local searches proposed in Section 5 in terms of solution quality. Each of the 600 available instances has been used to solve the four problems addressed in the paper with the three algorithms. For the problems with semi-active schedules, the proposed algorithms have been compared to the state-of-the-

art NEH heuristics: the CWT_w problems are solved with the NEH of Maassen et al. (2019), while the CIT_w with that of Liu et al. (2016).

Parameters. Each instance-problem-algorithm has been run with a time limit proportional to the size of the problem (number of jobs n and of machines m); it is computed as $\theta = \frac{60nm}{1000}$ (Balogh et al., 2022; Riahi et al., 2020), and measured in seconds (note that the NEH algorithms are constructive heuristics, thus no time limit is set in these cases). The effect of the time limit value has been analysed in Appendix A, which show detailed results of the performance of the proposed algorithms with various time limit values.

For the sliding window algorithm, after preliminary tests, the window size w_P has been set to assure the neighbourhood exploration to be run in, on average, less than 10 seconds. The resulting w_P values are: 8 jobs if $m \leq 20$, 6 if $20 < m \leq 40$, 4 if $m > 40$; the window size is reduced for larger values of m as the problem complexity increases; also, a time limit of 10 seconds has been set for all the MILP-based neighborhood explorations.

Results. Tables 4 and 5 show the results of the experiment in terms of ARPD values for the algorithms: sliding window (SW), one-opt and swap local searches. ARPD values are shown also for the state-of-the-art NEHs (by Maassen et al. (2019) and Liu et al. (2016)) for the problems with semi-active schedules. Each row displays the ARPD values of each algorithm for each problem version: the first part of each table displays the results for the VRF instances, and the last part for the Taillard benchmark. Each value is the average RPD among the 10 instances available for the combination n, m for each benchmark. The second-last row of each table shows the ARPD for each algorithm over all the VRF and Taillard instances, divided by small

Problem		General						Semi-active							
		SW		One-opt		Swap		SW		One-opt		Swap		NEH	
S	L	S	L	S	L	S	L	S	L	S	L	S	L	S	L
VRF instances															
10, 5	100, 20	**0.0**	**0.0**	33.7	18.2	22.4	16.5	**0.0**	**0.0**	86.4	73.3	56.0	52.1	106.1	77.3
10, 10	100, 40	**0.0**	**0.0**	31.1	33.8	25.4	32.2	**0.0**	**0.0**	99.3	47.0	60.2	37.5	113.4	49.4
10, 15	100, 60	**0.0**	**0.0**	33.1	25.0	24.4	23.1	**0.0**	**0.0**	92.1	28.6	59.1	22.6	99.1	30.0
10, 20	200, 20	**0.0**	**0.0**	30.0	7.6	22.8	7.0	**0.0**	**0.0**	60.0	37.8	43.2	34.4	68.8	40.1
20, 5	200, 40	**0.0**	**0.0**	42.7	18.3	34.8	17.1	**0.0**	**0.0**	146.6	30.9	107.6	28.2	171.4	33.0
20, 10	200, 60	**0.0**	**0.0**	38.9	13.9	35.3	13.1	**0.0**	**0.0**	128.3	20.0	100.2	18.2	142.8	21.2
20, 15	300, 20	**0.0**	**0.0**	41.7	4.7	32.5	4.2	**0.0**	**0.0**	99.8	22.2	74.5	20.0	112.4	24.4
20, 20	300, 40	**0.0**	**0.0**	39.5	9.6	33.8	9.0	**0.0**	**0.0**	92.5	25.9	68.8	24.0	100.8	27.6
30, 5	300, 60	**0.0**	**0.0**	38.6	7.0	33.7	6.6	**0.0**	**0.0**	157.5	13.4	121.9	12.6	177.6	14.7
30, 10	400, 20	**0.0**	**0.0**	39.3	3.5	34.6	3.0	**0.0**	**0.0**	146.5	12.3	101.2	10.5	159.3	14.1
30, 15	400, 40	**0.0**	**0.0**	42.6	6.2	37.8	5.7	**0.0**	**0.0**	117.1	14.1	90.8	13.1	124.0	15.4
30, 20	400, 60	**0.0**	**0.0**	41.5	4.6	36.5	4.1	**0.0**	**0.0**	97.4	7.4	75.6	6.6	106.1	8.4
40, 5	500, 20	**0.0**	**0.0**	36.8	3.4	32.3	3.2	**0.0**	**0.0**	197.8	11.6	133.1	9.8	217.6	12.3
40, 10	500, 40	**0.0**	**0.0**	39.7	5.0	35.4	4.8	**0.0**	**0.0**	146.5	8.9	114.8	7.6	159.9	9.4
40, 15	500, 60	**0.0**	**0.0**	37.5	2.4	32.3	2.0	**0.0**	**0.0**	138.3	6.9	97.3	5.8	150.9	7.4
40, 20	600, 20	**0.0**	**0.0**	41.8	2.8	36.6	2.6	**0.0**	**0.0**	110.3	9.0	89.5	7.5	118.6	9.9
50, 5	600, 40	**0.0**	**0.0**	34.3	5.2	30.8	5.0	**0.0**	**0.0**	214.5	5.7	154.5	4.8	237.7	6.7
50, 10	600, 60	**0.0**	**0.0**	37.2	2.5	32.0	2.1	**0.0**	**0.0**	153.2	6.2	109.7	5.2	160.7	6.7
50, 15	700, 20	**0.0**	**0.0**	39.2	2.6	35.7	2.4	**0.0**	**0.0**	129.1	9.2	104.5	7.8	140.9	10.1
50, 20	700, 40	**0.0**	**0.0**	39.6	2.3	34.4	2.1	**0.0**	**0.0**	92.4	5.5	76.4	4.4	98.9	6.0
60, 5	700, 60	**0.0**	**0.0**	33.9	1.8	31.5	1.6	**0.0**	**0.0**	202.8	5.3	170.8	4.3	221.0	5.6
60, 10	800, 20	**0.0**	**0.0**	38.2	3.4	35.1	3.2	**0.0**	**0.0**	173.6	7.3	140.8	5.6	188.8	7.8
60, 15	800, 40	**0.0**	**0.0**	32.6	1.5	30.6	1.2	**0.0**	**0.0**	125.8	5.2	96.2	4.2	135.2	5.8
60, 20	800, 60	**0.0**	**0.0**	36.8	1.7	34.7	1.4	**0.0**	**0.0**	91.7	4.8	74.8	3.7	97.6	5.0
Taillard instances															
20, 5	100, 5	**0.0**	**0.0**	36.2	34.5	30.1	32.2	**0.0**	**0.0**	127.4	277.3	90.7	214.4	152.2	296.7
20, 10	100, 10	**0.0**	**0.0**	39.6	32.2	30.2	30.4	**0.0**	**0.0**	121.5	184.0	94.3	153.8	131.8	191.3
20, 20	100, 20	**0.0**	**0.0**	38.8	20.5	31.4	18.9	**0.0**	**0.0**	86.8	76.7	64.0	60.2	98.7	80.0
50, 5	200, 10	**0.0**	**0.0**	35.5	13.1	31.9	12.4	**0.0**	**0.0**	192.3	101.7	135.2	90.9	206.3	104.2
50, 10	200, 20	**0.0**	**0.0**	41.7	8.2	35.7	7.6	**0.0**	**0.0**	165.5	46.8	122.6	41.7	173.6	49.3
50, 20	500, 20	**0.0**	**0.0**	37.8	3.2	33.9	2.9	**0.0**	**0.0**	108.8	10.2	83.8	8.5	117.1	11.3
Overall		**0.0**	**0.0**	41.3	9.9	35.8	9.3	**0.0**	**0.0**	142.5	37.1	108.4	30.6	143.0	39.3
α		100 %		90%		100%		100%		96%		100%			

Table 4: Average RPD of each algorithm for the problems related to CWT_w (%). The algorithm NEH refers to Maassen et al. (2019).

(i.e., with less than 100 jobs - problem 'S' in the tables) and large (i.e., with at least 100 jobs - problem 'L' in the tables) instances. The last row of each table shows the value of the percentage of instances in which each local search is able to improve the initial solution given by the NEH (i.e., the α value).

For each version of the problem, among the three local searches, the sliding window algorithm achieves the best performance (i.e., it always has

Problem		General						Semi-active							
		SW		**One-opt**		**Swap**		**SW**		**One-opt**		**Swap**		**NEH**	
S	L	S	L	S	L	S	L	S	L	S	L	S	L	S	L
VRF instances															
10, 5	100, 20	**0.0**	**0.0**	10.3	9.9	5.8	8.3	**0.0**	**0.0**	1.9	10.1	1.9	10.4	1.9	10.4
10, 10	100, 40	**0.0**	**0.0**	11.9	11.9	6.1	10.6	**0.0**	**0.0**	4.5	5.6	4.7	5.6	4.7	5.6
10, 15	100, 60	**0.0**	**0.0**	12.1	6.9	8.4	5.5	**0.0**	**0.0**	8.0	1.9	8.0	2.0	8.0	2.0
10, 20	200, 20	**0.0**	**0.0**	10.2	3.9	7.2	2.8	**0.0**	**0.0**	8.8	4.2	9.1	4.3	9.1	4.3
20, 5	200, 40	**0.0**	**0.0**	7.5	4.2	5.4	3.3	**0.0**	**0.0**	6.4	4.1	6.7	4.1	6.7	4.1
20, 10	200, 60	**0.0**	**0.0**	13.8	3.1	9.9	2.2	**0.0**	**0.0**	12.2	2.0	13.8	2.1	13.8	2.1
20, 15	300, 20	**0.0**	**0.0**	13.9	2.3	10.4	1.4	**0.0**	**0.0**	10.2	2.2	10.3	2.3	10.3	2.3
20, 20	300, 40	**0.0**	**0.0**	15.2	1.8	14.0	1.3	**0.0**	**0.0**	12.3	1.9	12.6	1.9	10.1	1.9
30, 5	300, 60	**0.0**	**0.0**	5.5	1.7	3.3	1.0	**0.0**	**0.0**	3.9	1.2	3.9	1.2	3.9	1.2
30, 10	400, 20	**0.0**	**0.0**	13.8	1.5	10.0	1.0	**0.0**	**0.0**	12.5	1.1	12.6	1.1	12.6	1.1
30, 15	400, 40	**0.0**	**0.0**	15.9	1.6	13.8	1.1	**0.0**	**0.0**	13.6	1.1	13.5	1.2	13.9	1.2
30, 20	400, 60	**0.0**	**0.0**	16.7	1.1	14.1	0.6	**0.0**	**0.0**	12.7	0.7	12.7	0.8	12.7	0.8
40, 5	500, 20	**0.0**	**0.0**	4.8	1.3	2.3	1.0	**0.0**	**0.0**	2.7	0.8	2.6	0.8	2.8	0.8
40, 10	500, 40	**0.0**	**0.0**	10.2	1.3	7.5	0.7	**0.0**	**0.0**	8.6	0.6	8.6	0.7	8.6	0.7
40, 15	500, 60	**0.0**	**0.0**	16.0	0.8	13.2	0.4	**0.0**	**0.0**	13.7	0.4	13.8	0.5	13.9	0.5
40, 20	600, 20	**0.0**	**0.0**	18.5	1.1	16.7	0.6	**0.0**	**0.0**	11.4	0.6	12.3	0.7	12.3	0.7
50, 5	600, 40	**0.2**	**0.0**	4.9	0.9	2.4	0.4	**0.0**	**0.0**	1.3	0.3	1.2	0.3	1.3	0.3
50, 10	600, 60	**0.0**	**0.0**	9.6	0.6	6.7	0.2	**0.0**	**0.0**	8.8	0.4	8.8	0.4	8.8	0.4
50, 15	700, 20	**0.0**	**0.0**	14.6	0.9	12.4	0.5	**0.0**	**0.0**	14.2	0.4	14.3	0.4	14.3	0.4
50, 20	700, 40	**0.0**	**0.0**	17.4	0.7	14.8	0.3	**0.0**	**0.0**	13.5	0.4	13.6	0.4	13.6	0.4
60, 5	700, 60	**0.0**	**0.0**	4.5	0.5	2.8	0.0	**0.0**	**0.0**	2.9	0.3	2.2	0.4	3.0	0.4
60, 10	800, 20	**0.0**	**0.0**	7.9	1.0	5.7	0.6	**0.0**	**0.0**	6.7	0.3	6.8	0.3	6.8	0.3
60, 15	800, 40	**0.0**	**0.0**	12.8	0.5	10.9	0.1	**0.0**	**0.0**	14.6	0.3	14.7	0.3	14.7	0.3
60, 20	800, 60	**0.0**	**0.0**	16.1	0.4	13.2	**0.0**	**0.0**	**0.0**	12.8	0.3	13.1	0.3	13.1	0.4
Taillard instances															
20, 5	100, 5	**0.0**	**0.0**	8.8	1.8	5.8	0.5	**0.0**	**0.0**	5.5	0.5	5.2	0.5	5.8	0.5
20, 10	100, 10	**0.0**	**0.0**	12.7	4.9	9.3	3.0	**0.0**	**0.0**	13.5	2.2	13.5	2.2	9.3	3.0
20, 20	100, 20	**0.0**	**0.0**	16.1	9.6	12.6	7.7	**0.0**	**0.0**	13.0	8.3	13.2	8.4	12.6	7.7
50, 5	200, 10	**0.0**	**0.0**	3.4	2.1	1.2	1.1	**0.0**	**0.0**	1.3	1.2	0.8	1.2	1.3	1.1
50, 10	200, 20	**0.0**	**0.0**	8.8	3.8	6.1	2.8	**0.0**	**0.0**	6.9	4.1	6.9	4.2	6.9	2.8
50, 20	500, 20	**0.0**	**0.0**	16.3	1.2	14.0	0.7	**0.0**	**0.0**	10.7	0.7	10.9	0.7	14.0	0.7
Overall		**0.0**	**0.0**	15.0	2.8	12.1	2.0	**0.0**	**0.0**	15.0	1.9	15.1	2.0	12.2	1.9
α		100 %		73%		100%		97%		20%		3%			

Table 5: Average RPD of each algorithm for the problems related to CIT_w (%). The algorithm NEH refers to Liu et al. (2016).

a zero ARPD), while there is no significant difference between the one-opt and swap algorithms (an Anova test has been used to detect it). Also non parametric tests such as the Kruskal-Wallis (Corder and Foreman, 2014) confirmed the statistical difference between the performance of the sliding window with respect to the other algorithms, for all the problem versions. For small problems, the sliding windows largely outperforms the other algorithms

for each problem version, while for large problems the average RPDs for the one-opt and swap algorithms notably decrease. Also, by comparing the two tables, the difference between SW and the other two local searches is larger for the CWT_w problem than for the CIT_w; this can be motivated by the fact that the sliding window local search is able to explore a larger portion of solution space to find good solutions for the CWT_w problem, while for the CIT_w all the local searches perform similarly (however with a positive statistical difference for the SW). While there is no appreciable difference in the behavior of each algorithm in the cases of general and semi-active schedules for the CIT_w problem (see the Overall results of Table 5), a largely different performance is achieved by each of the three searches for the CWT_w problem when moving from general to semi-active schedules (see the Overall results of Table 4).

To compare the proposed algorithms with the state-of-the-art NEH heuristics, the ARPD and α values are available in the two tables. For all the problem versions, the SW local search is able to improve the NEH algorithm in 2384 instances out of 2400 (i.e., 600 available instances times 4 problem versions). More specifically, no improvement from the NEH is achieved in 16 instances for the CIT_w semi-active problem. Indeed, this problem version results the most difficult to solve from Table 5. For the CIT_w problem, the one-opt and swap searches could improve the NEH initial solution only in 122 and 20 instances out of 600, respectively (i.e., α equal to 20% and 3%). All in all, the NEH state-of-the-art heuristics are largely outperformed by the proposed SW local search for all the problem versions. The one-opt and swap algorithms outperform the state-of-the-art in all the problem versions

but the CIT_w semi-active.

7. Conclusions and future works

This paper addressed four variants of the permutation flow shop scheduling problem. Two objective functions were studied: the weighted sum of core waiting time and makespan, and the weighted sum of core idle time and makespan. For each of them, both the cases of semi-active and general schedules were considered. A unique and general framework was proposed to consider alternatively these four problem variants. Within the framework, three formalization models were proposed and tested (i.e., a MILP with positional variables, a MILP with precedence variables, and a Constraint Programming model). Also, three local searches were developed, each based on a different neighborhood. The framework was tested over two benchmark data-sets available in the literature (600 instances at all for each problem variant), and compared with the state-of-the-art solving algorithms.

The results showed that the MILP with positional variables is the more efficient formalization model for all the problem variants. Also, the proposed local searches are competitive with respect to the state-of-the-art algorithms available in the literature. More specifically, the proposed sliding window algorithm largely outperforms the state-of-the-art in almost all the cases.

Future research will involve the design of ad-hoc exact and metaheuristic approaches for each of the problems studied in this paper, which may take advantage of problem-specific properties. Regarding exact approaches, the study of combinatorial bounds and/or dominance rules may be worthwhile to explore in order to strengthen the MILP/CP formulations and/or design

efficient branching algorithm based on memorization. Regarding metaheuristics, the definition of speed up techniques for the computation of the objective function seems to be a key point for achieving high quality results. Moreover, as the approach proposed in this paper is able to address both semi-active and general schedules, and different objective functions, it can be further improved to handle multi-objective problems.

Appendix A. Time limit analysis

The developed algorithms (sliding window, one-opt and swap local searches) are tested to assess the performance with various time limits. For this experiment, the Taillard and VRF instances are used (as in Section 6) and, for each combination of number of jobs and machines, five out of the available 10 instances are tested for all the algorithms and for the four problem variants. In total, 300 instances are considered.

To check how the results obtained by the three algorithms change with various time limits, three time limit values are tested. As in Section 6.2, the time limits depend on the numbers of jobs n and of machines m, and the following values are considered: $\theta = \frac{60nm}{1000}$ (as in 6.2), $\frac{\theta}{2}, \frac{\theta}{4}$. For each instance-problem-algorithm (defined by index i), the solutions obtained in the three time limits are compared in terms of Percentage Difference (PD) as:

$$PD_{i,\frac{\theta}{2}} = \frac{ub_{i,\frac{\theta}{2}} - ub_{i,\theta}}{ub_{i,\theta}}, \quad PD_{i,\frac{\theta}{4}} = \frac{ub_{i,\frac{\theta}{4}} - ub_{i,\theta}}{ub_{i,\theta}}.$$

Tables A.6 and A.7 show the results in terms of Average PD (APD), i.e., the average of PD values over the five instances for the CWT_w and the CIT_w

problems, respectively.

The results of Tables A.6 and A.7 show that the one-opt and swap local searches are poorly affected by different time limits. In fact, especially for the semi-active problem variants with a small number of jobs and machines, they rarely reach the time limit. Instead, for larger instances, they have positive percentage gap that increases with the decrease of the time limit value (from $\frac{\theta}{2}$ to $\frac{\theta}{4}$). The sliding window search, instead, is more affected by the time limit, and the APD values are positive also for small instances.

	General						**Semi active**					
	SW		**One-opt**		**Swap**		**SW**		**One-opt**		**Swap**	
instance	$APD_{\frac{\theta}{2}}$	$APD_{\frac{\theta}{4}}$	$APD_{\frac{\theta}{2}}$	$APD_{\frac{\theta}{4}}$	$APD_{\frac{\theta}{2}}$	$APD_{\frac{\theta}{4}}$	$APD_{\frac{\theta}{2}}$	$APD_{\frac{\theta}{4}}$	$APD_{\frac{\theta}{2}}$	$APD_{\frac{\theta}{4}}$	$APD_{\frac{\theta}{2}}$	$APD_{\frac{\theta}{4}}$
VRF instances												
10, 5	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10, 10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0
10, 15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10, 20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
20, 5	0.9	6.7	0.0	0.0	0.0	0.0	3.1	11.2	0.0	0.0	0.0	0.0
20, 10	1.0	6.2	0.0	0.0	0.0	0.0	2.0	7.8	0.0	0.0	0.0	0.0
20, 15	4.6	8.0	0.0	0.0	0.0	0.0	0.3	9.3	0.0	0.0	0.0	0.0
20, 20	2.1	7.8	0.0	0.0	0.0	0.0	3.2	17.4	0.0	0.0	0.0	0.0
30, 5	2.9	7.5	0.0	0.0	0.0	0.0	7.6	16.8	0.0	0.0	0.0	0.0
30, 10	3.7	11.5	0.0	0.0	0.0	0.0	8.1	21.2	0.0	0.0	0.0	0.0
30, 15	5.7	16.0	0.0	0.0	0.0	0.0	7.7	24.7	0.0	0.0	0.0	0.0
30, 20	7.1	16.8	0.0	0.0	0.0	0.0	10.0	41.1	0.0	0.0	0.0	0.0
40, 5	2.4	11.9	0.0	0.0	0.0	0.0	5.1	27.2	0.0	0.0	0.0	0.0
40, 10	4.5	17.3	0.0	0.0	0.0	0.0	6.6	31.8	0.0	0.0	0.0	0.0
40, 15	4.1	14.1	0.0	0.0	0.0	0.0	11.9	57.0	0.0	0.0	0.0	0.0
40, 20	6.0	16.8	0.0	0.0	0.0	0.0	11.1	45.9	0.0	0.0	0.0	0.0
50, 5	5.2	14.9	0.0	0.0	0.0	0.0	8.2	27.9	0.0	0.0	0.0	0.0
50, 10	5.5	18.1	0.0	0.0	0.0	0.0	10.1	40.2	0.0	0.0	0.0	0.0
50, 15	7.9	17.4	0.0	0.0	0.0	0.0	16.6	59.4	0.0	0.0	0.0	0.0
50, 20	11.1	22.2	0.0	0.0	0.0	0.0	14.2	43.3	0.0	0.0	0.0	0.0
60, 5	7.1	16.5	0.0	0.0	0.0	0.0	10.0	77.8	0.0	0.0	0.0	0.0
60, 10	10.4	21.9	0.0	0.0	0.0	0.0	24.6	63.7	0.0	0.0	0.0	0.0
60, 15	12.4	19.1	0.0	0.0	0.0	0.0	19.1	62.8	0.0	0.0	0.0	0.0
60, 20	13.7	21.1	0.0	0.0	0.0	0.0	25.1	47.3	0.0	0.0	0.0	0.0
100, 20	8.3	11.4	0.0	0.0	0.0	0.0	31.2	49.7	0.0	0.0	0.0	0.0
100, 40	12.2	21.3	0.0	0.0	0.0	0.0	8.4	20.4	0.0	0.0	0.0	0.0
100, 60	7.9	17.5	0.0	0.0	0.0	0.0	4.9	11.3	0.0	0.0	0.0	0.0
200, 20	2.9	4.5	0.0	0.0	0.0	0.0	18.5	24.9	0.0	0.0	0.0	0.0
200, 40	8.7	12.2	0.0	0.0	0.0	0.0	12.6	20.9	0.0	0.0	0.0	0.0
200, 60	8.5	11.5	0.0	0.0	0.0	0.0	6.7	11.4	0.0	0.0	0.0	0.0
300, 20	1.8	2.7	0.0	0.0	0.0	0.0	12.4	16.4	0.0	0.0	0.0	0.0

300, 40	5.4	7.1	0.0	0.0	0.0	0.0	11.5	16.2	0.0	0.5	0.2	0.4
300, 60	3.8	5.2	0.0	0.0	0.0	0.0	5.4	8.4	0.0	0.6	0.0	0.2
400, 20	1.8	2.1	0.0	0.0	0.0	0.0	6.8	9.3	0.0	0.4	0.3	0.4
400, 40	2.8	3.8	0.0	0.0	0.0	0.0	6.4	9.3	0.1	0.7	0.1	0.3
400, 60	2.6	3.6	0.0	0.0	0.0	0.0	2.8	5.5	0.4	0.6	0.1	0.2
500, 20	2.0	2.4	0.0	0.0	0.0	0.0	5.5	7.0	0.1	0.2	0.2	0.3
500, 40	2.7	3.7	0.0	0.0	0.0	0.0	5.2	7.2	0.2	0.3	0.2	0.3
500, 60	0.1	1.2	0.0	0.0	0.0	0.0	3.2	5.1	0.1	0.3	0.0	0.0
600, 20	1.6	2.0	0.0	0.0	0.0	0.1	4.5	6.8	0.1	0.3	0.1	0.2
600, 40	2.8	3.9	0.0	0.0	0.0	0.1	2.0	5.0	0.2	0.5	0.3	0.4
600, 60	0.4	1.2	0.0	0.0	0.0	0.0	3.7	5.2	0.2	0.3	0.3	0.4
700, 20	1.5	1.8	0.0	0.0	0.0	0.0	5.6	6.4	0.3	0.6	0.2	0.5
700, 40	0.2	0.9	0.0	0.0	0.0	0.0	2.4	3.6	0.2	0.2	0.2	0.4
700, 60	0.4	1.0	0.0	0.0	0.0	0.0	3.7	4.3	0.1	0.2	0.2	0.3
800, 20	2.2	2.5	0.0	0.0	0.0	0.0	4.3	5.2	0.3	0.4	0.2	0.3
800, 40	0.1	0.5	0.0	0.0	0.0	0.0	3.2	4.0	0.2	0.4	0.3	0.4
800, 60	0.9	1.2	0.0	0.0	0.0	0.0	3.4	3.9	0.1	0.2	0.1	0.2
Taillard instances												
20, 5	0.5	3.1	0.0	0.0	0.0	0.0	1.9	5.4	0.0	0.0	0.0	0.0
20, 10	2.2	4.8	0.0	0.0	0.0	0.0	2.0	6.2	0.0	0.0	0.0	0.0
20, 20	0.9	4.8	0.0	0.0	0.0	0.0	1.9	8.9	0.0	0.0	0.0	0.0
50, 5	5.1	15.6	0.0	0.0	-0.4	0.7	7.3	53.3	0.0	0.0	1.4	4.2
50, 10	9.6	18.7	0.0	0.0	0.1	0.2	16.3	73.7	0.0	0.0	0.0	1.7
50, 20	13.1	22.6	0.0	0.0	0.3	0.6	22.5	48.8	0.0	0.0	0.0	-0.1
100, 5	11.0	21.8	0.0	0.0	0.4	0.5	33.4	129.5	0.0	0.0	2.5	6.0
100, 10	15.3	21.2	0.0	0.0	0.6	0.8	33.6	107.0	0.0	0.0	3.2	10.6
100, 20	8.4	11.5	0.0	0.0	0.3	0.5	27.2	44.1	0.0	0.0	1.5	3.1
200, 10	7.0	9.7	0.0	0.0	0.0	0.0	49.6	76.1	0.0	0.0	0.8	1.0
200, 20	3.3	4.9	0.0	0.0	0.0	0.1	22.6	30.4	0.0	0.0	0.8	1.1
500, 20	1.8	2.2	0.0	0.0	0.1	0.1	5.0	6.8	0.0	1.0	0.1	0.0

Table A.6: APD values of each algorithm for different time limits for the problems related to CWT_w (%)

	General						**Semi active**					
	SW		**One-opt**		**Swap**		**SW**		**One-opt**		**Swap**	
instance	$APD_{\frac{\theta}{2}}$	$APD_{\frac{\theta}{4}}$	$APD_{\frac{\theta}{2}}$	$APD_{\frac{\theta}{4}}$	$APD_{\frac{\theta}{2}}$	$APD_{\frac{\theta}{4}}$	$APD_{\frac{\theta}{2}}$	$APD_{\frac{\theta}{4}}$	$APD_{\frac{\theta}{2}}$	$APD_{\frac{\theta}{4}}$	$APD_{\frac{\theta}{2}}$	$APD_{\frac{\theta}{4}}$
VRF instances												
10, 5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10, 10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.0	0.0	0.0
10, 15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10, 20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
20, 5	0.0	0.6	0.0	0.0	0.0	0.0	0.0	1.1	0.0	0.0	0.0	0.0
20, 10	0.0	0.7	0.0	0.0	0.0	0.0	0.4	1.1	0.0	0.0	0.0	0.0
20, 15	0.0	1.0	0.0	0.0	0.0	0.0	0.9	2.3	0.0	0.0	0.0	0.0
20, 20	0.4	2.3	0.0	0.0	0.0	0.0	1.4	2.7	0.0	0.0	0.0	0.0
30, 5	0.5	0.8	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0
30, 10	0.0	2.1	0.0	0.0	0.0	0.0	1.5	6.4	0.0	0.0	0.0	0.0
30, 15	0.9	3.3	0.0	0.0	0.0	0.0	0.6	2.8	0.0	0.0	0.0	0.0
30, 20	0.2	2.6	0.0	0.0	0.0	0.0	1.7	3.5	0.0	0.0	0.0	0.0

40, 5	0.0	0.2	0.0	0.0	0.0	0.0	0.1	0.3	0.0	0.0	0.0	0.0
40, 10	1.4	3.4	0.0	0.0	0.0	0.0	0.5	2.6	0.0	0.0	0.0	0.0
40, 15	1.0	3.6	0.0	0.0	0.0	0.0	1.5	4.4	0.0	0.0	0.0	0.0
40, 20	2.0	5.9	0.0	0.0	0.0	0.0	1.9	6.1	0.0	0.0	0.0	0.0
50, 5	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
50, 10	0.9	2.1	0.0	0.0	0.0	0.0	1.9	4.4	0.0	0.0	0.0	0.0
50, 15	2.0	5.7	0.0	0.0	0.0	0.0	2.1	5.4	0.0	0.0	0.0	0.0
50, 20	2.9	5.4	0.0	0.0	0.0	0.0	3.0	7.8	0.0	0.0	0.0	0.0
60, 5	0.4	1.2	0.0	0.0	0.0	0.0	0.4	1.1	0.0	0.0	0.0	0.0
60, 10	0.6	1.1	0.0	0.0	0.0	0.0	1.1	2.1	0.0	0.0	0.0	0.0
60, 15	2.5	5.8	0.0	0.0	0.0	0.0	3.3	8.0	0.0	0.0	0.0	0.0
60, 20	4.1	7.4	0.0	0.0	0.0	0.0	5.1	10.0	0.0	0.0	0.0	0.0
100. 20	3.3	4.6	0.0	0.0	0.0	0.0	3.0	6.9	0.0	0.0	0.0	0.0
100. 40	5.6	7.7	0.0	0.0	0.0	0.0	1.5	3.6	0.0	0.0	0.0	0.0
100. 60	2.6	4.3	0.0	0.0	0.0	0.0	0.5	1.4	0.0	0.0	0.0	0.0
200. 20	0.9	1.3	0.0	0.1	0.0	0.0	2.7	3.0	0.0	0.0	0.0	0.0
200. 40	1.5	2.3	0.0	0.0	0.0	0.0	2.5	3.3	0.0	0.0	0.0	0.0
200. 60	1.3	1.9	0.0	0.1	0.0	0.0	1.1	1.5	0.0	0.0	0.0	0.0
300. 20	0.3	0.7	0.0	0.1	0.0	0.0	1.7	1.9	0.0	0.0	0.0	0.0
300. 40	0.4	0.7	0.0	0.1	0.0	0.0	1.1	1.4	0.0	0.0	0.0	0.0
300. 60	0.5	0.9	0.0	0.0	0.0	0.0	0.7	0.9	0.0	0.0	0.0	0.0
400. 20	0.4	0.5	0.2	0.2	0.0	0.0	0.4	0.6	0.0	0.0	0.0	0.0
400. 40	0.4	0.7	0.0	0.1	0.0	0.0	0.3	0.8	0.0	0.0	0.0	0.0
400. 60	0.5	0.8	0.2	0.2	0.0	0.0	0.5	0.6	0.1	0.1	0.0	0.0
500. 20	0.2	0.4	0.1	0.1	0.0	0.0	0.4	0.5	0.0	0.0	0.0	0.0
500. 40	0.5	0.6	0.1	0.1	0.0	0.0	0.2	0.3	0.0	0.1	0.0	0.0
500. 60	0.3	0.5	0.0	0.1	0.0	0.0	0.2	0.3	0.0	0.0	0.0	0.0
600. 20	0.1	0.2	0.0	0.0	0.0	0.2	0.2	0.3	0.0	0.0	0.0	0.0
600. 40	0.2	0.4	0.0	0.1	0.0	0.0	-0.1	0.0	0.0	0.0	0.0	0.0
600. 60	0.0	0.3	0.1	0.1	0.0	0.0	0.2	0.3	0.0	0.0	0.0	0.0
700. 20	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0
700. 40	0.0	0.3	0.0	0.0	0.0	0.0	0.1	0.2	0.0	0.0	0.0	0.0
700. 60	-0.1	0.1	0.0	0.0	0.0	0.0	0.2	0.2	0.0	0.0	0.0	0.0
800. 20	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
800. 40	-0.2	0.0	0.0	0.0	0.0	0.0	-0.1	0.1	0.0	0.0	0.0	0.0
800. 60	-0.3	-0.1	0.0	0.0	0.0	0.0	0.2	0.2	0.0	0.0	0.0	0.0
Taillard instances												
20, 5	0.8	1.0	0.0	0.0	0.0	0.0	0.0	0.3	0.0	0.0	0.0	0.0
20, 10	0.0	1.2	0.0	0.0	0.0	0.0	0.0	1.4	0.0	0.0	0.0	0.0
20, 20	0.7	1.8	0.0	0.0	0.0	0.0	1.4	3.8	0.0	0.0	0.0	0.0
50, 5	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
50, 10	0.9	2.4	0.0	0.0	0.0	0.0	1.8	4.3	0.0	0.0	0.0	0.0
50, 20	3.1	5.3	0.0	0.0	0.0	0.4	2.6	5.7	0.0	0.0	0.0	0.0
100, 5	0.0	0.1	0.0	0.0	0.0	0.0	0.2	0.2	0.0	0.0	0.0	0.0
100, 10	0.6	1.3	0.0	0.0	0.0	0.1	0.4	1.0	0.0	0.0	0.0	0.0
100, 20	2.8	4.4	0.0	0.0	0.2	0.6	2.5	5.0	0.0	0.0	0.0	0.0
200, 10	1.0	1.1	0.0	0.0	0.1	0.2	0.4	0.6	0.0	0.0	0.0	0.0
200, 20	0.9	1.5	0.0	0.1	0.3	0.3	2.1	2.7	0.0	0.0	0.0	0.0
500, 20	0.1	0.3	0.1	0.2	0.2	0.3	0.2	0.3	0.0	0.0	0.0	0.0

Table A.7: APD values of each algorithm for different time limits for the problems related to CIT_w (%)

References

Allahverdi, A., 2016. A survey of scheduling problems with no-wait in process. European Journal of Operational Research 255, 665–686.

Allahverdi, A., Aydilek, H., Aydilek, A., 2020. No-wait flowshop scheduling problem with separate setup times to minimize total tardiness subject to makespan. Applied Mathematics and Computation 365, 124688.

Bagchi, U., 1989. Simultaneous minimization of mean and variation of flow time and waiting time in single machine systems. Operations Research 37, 118–125.

Balogh, A., Garraffa, M., O'Sullivan, B., Salassa, F., 2022. Milp-based local search procedures for minimizing total tardiness in the no-idle permutation flowshop problem. Computers & Operations Research , 105862.

Bektaş, T., Hamzadayı, A., Ruiz, R., 2020. Benders decomposition for the mixed no-idle permutation flowshop scheduling problem. Journal of Scheduling 23, 513–523.

Birgin, E.G., Ferreira, J.E., Ronconi, D.P., 2020. A filtered beam search method for the m-machine permutation flowshop scheduling problem minimizing the earliness and tardiness penalties and the waiting time of the jobs. Computers & Operations Research 114, 104824.

Chu, C., Portmann, M.C., 1993. Job-shop scheduling to minimize total waiting time. Applied stochastic models and data analysis 9, 177–185.

Corder, G.W., Foreman, D.I., 2014. Nonparametric statistics: A step-by-step approach. John Wiley & Sons.

De Abreu, A.P., Fuchigami, H.Y., 2022. An efficiency and robustness analysis of warm-start mathematical models for idle and waiting times optimization in the flow shop. Computers & Industrial Engineering 166, 107976.

De Fátima Morais, M., Ribeiro, M.H.D.M., da Silva, R.G., Mariani, V.C., dos Santos Coelho, L., 2022. Discrete differential evolution metaheuristics for permutation flow shop scheduling problems. Computers & Industrial Engineering 166, 107956.

Eilon, S., Chowdhury, I., 1977. Minimising waiting time variance in the single machine problem. Management Science 23, 567–575.

van Essen, J.T., Hans, E.W., Hurink, J.L., Oversberg, A., 2012. Minimizing the waiting time for emergency surgery. Operations Research for Health Care 1, 34–44.

Framinan, J.M., Leisten, R., 2008. Total tardiness minimization in permutation flow shops: a simple approach based on a variable greedy algorithm. International Journal of Production Research 46, 6479–6498.

Goncharov, Y., Sevastyanov, S., 2009. The flow shop problem with no-idle constraints: A review and approximation. European Journal of Operational Research 196, 450–456.

Höhn, W., Jacobs, T., Megow, N., 2012. On eulerian extensions and their application to no-wait flowshop scheduling. Journal of Scheduling 15, 295–309.

Hosseini, N., Tavakkoli-Moghaddam, R., 2013. Two meta-heuristics for solving a new two-machine flowshop scheduling problem with the learning effect and dynamic arrivals. The International Journal of Advanced Manufacturing Technology 65, 771–786.

Kubiak, W., 1993. Completion time variance minimization on a single machine is difficult. Operations Research Letters 14, 49–59.

Laborie, P., Rogerie, J., Shaw, P., Vilím, P., 2018. Ibm ilog cp optimizer for scheduling. Constraints 23, 210–250.

Lees-Miller, J.D., 2016. Minimising average passenger waiting time in personal rapid transit systems. Annals of Operations Research 236, 405–424.

Li, X., Ye, N., Xu, X., Sawhey, R., 2007. Influencing factors of job waiting time variance on a single machine. European Journal of Industrial Engineering 1, 56–73.

Liao, C.J., Tseng, C.T., Luarn, P., 2007. A discrete version of particle swarm optimization for flowshop scheduling problems. Computers & Operations Research 34, 3099–3111.

Little, J.D., 1961. A proof for the queuing formula: L= λ w. Operations research 9, 383–387.

Liu, W., Jin, Y., Price, M., 2016. A new nawaz–enscore–ham-based heuristic for permutation flow-shop problems with bicriteria of makespan and machine idle time. Engineering Optimization 48, 1808–1822.

Maassen, K., Hipp, A., Perez-Gonzalez, P., 2019. Constructive heuristics for the minimization of core waiting time in permutation flow shop problems, in: 2019 International Conference on Industrial Engineering and Systems Management (IESM), IEEE. pp. 1–6.

Maassen, K., Perez-Gonzalez, P., Günther, L.C., 2020. Relationship between common objective functions, idle time and waiting time in permutation flow shop scheduling. Computers & Operations Research 121, 104965.

Mao, J.Y., Pan, Q.K., Miao, Z.H., Gao, L., Chen, S., 2022. A hash map-based memetic algorithm for the distributed permutation flowshop scheduling problem with preventive maintenance to minimize total flow-time. Knowledge-Based Systems 242, 108413.

de Matta, R., 2019. Minimizing the total waiting time of intermediate products in a manufacturing process. International transactions in operational research 26, 1096–1117.

Merten, A.G., Muller, M.E., 1972. Variance minimization in single machine sequencing problems. Management Science 18, 518–528.

Nawaz, M., Enscore, E.E., Ham, I., 1983. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega 11, 91–95. URL: https://www.sciencedirect.com/science/article/pii/0305048383900889, doi:https://doi.org/10.1016/0305-0483(83)90088-9.

Pinedo, M.L., 2012. Scheduling. volume 29. Springer.

Riahi, V., Chiong, R., Zhang, Y., 2020. A new iterated greedy algorithm for

no-idle permutation flowshop scheduling with the total tardiness criterion. Computers & operations research 117, 104839.

Saber, R.G., Ranjbar, M., 2022. Minimizing the total tardiness and the total carbon emissions in the permutation flow shop scheduling problem. Computers & Operations Research 138, 105604.

Sharma, M., Sharma, S., Sharma, M., 2020. No-wait flowshop scheduling problem with bicriteria of idle time and makespan, in: Soft Computing: Theories and Applications. Springer, pp. 549–557.

Silva, A.F., Valente, J.M., Schaller, J.E., 2022. Metaheuristics for the permutation flowshop problem with a weighted quadratic tardiness objective. Computers Operations Research 140, 105691.

Soroush, H., 2012. Solving the single machine scheduling problem with general job-dependent past-sequence-dependent setup times and learning effects. European Journal of Industrial Engineering 6, 596–628.

Sun, L., Sun, L., Wang, J.B., 2011. Single-machine scheduling to minimize total absolute differences in waiting times with deteriorating jobs. Journal of the Operational Research Society 62, 768–775.

Szwarc, W., Mukhopadhyay, S.K., 1995. Minimizing a quadratic cost function of waiting times in single-machine scheduling. Journal of the Operational Research Society 46, 753–761.

Taillard, E., 1993. Benchmarks for basic scheduling problems. european journal of operational research 64, 278–285.

Vallada, E., Ruiz, R., Framinan, J.M., 2015. New hard benchmark for flowshop scheduling problems minimising makespan. European Journal of Operational Research 240, 666–677.

Xu, X., 2011. Minimizing weighted waiting time variance on a single processor. Computers & Industrial Engineering 61, 1233–1239.

Xu, X., Ye, N., 2007. Minimization of job waiting time variance on identical parallel machines. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 37, 917–927.

Yagmahan, B., Yenisey, M.M., 2008. Ant colony optimization for multi-objective flow shop scheduling problem. Computers & Industrial Engineering 54, 411–420.

Ye, N., Li, X., Farley, T., Xu, X., 2007. Job scheduling methods for reducing waiting time variance. Computers & Operations Research 34, 3069–3083.

Zhou, S., Cai, X., 1996. Variance minimization–relationship between completion-time variance and waiting-time variance. The ANZIAM Journal 38, 126–139.

Öztop, H., Tasgetiren, M.F., Eliiyi, D.T., Pan, Q.K., Kandiller, L., 2020. An energy-efficient permutation flowshop scheduling problem. Expert Systems with Applications 150, 113279.