An effective approach for total completion time minimization subject to makespan constraint in permutation flowshops

ARTICLE TEMPLATE

An effective approach for total completion time minimization subject to makespan constraint in permutation flowshops

E. Pastore[a] and A. Alfieri[a]

[a]Department of Management and Production Engineering, Politecnico di Torino, Turin, Italy

ARTICLE HISTORY
Compiled January 2, 2024

ABSTRACT
This article addresses the permutation flowshop scheduling problem with the objective of minimizing the total completion time subject to a makespan constraint. As the makespan is related to system utilization while the total completion time is related to the waiting time (and hence to the work in process (WIP)), in real contexts, focusing on both total completion time and makespan allows to find a good trade-off between WIP and utilization. Two local search algorithms are developed and, by using an extensive computational experience on literature benchmark instances, they are proved to be able to find good solutions both for regular and no-wait flowshops.

KEYWORDS
Scheduling; makespan; total completion time; permutation flowshop

AMS CLASSIFICATION
90C27; 90B35; 90C11

1. Introduction

Flowshops are among the most adopted layouts in production systems, especially in contexts with sufficiently large product volumes and low product variety.

When volumes are really large and no more than two or three very similar product types are manufactured, synchronous lines are used. In these systems, the production is paced by an interconnecting conveyor system. No work in process (WIP) can accumulate at the different stations and the production control logic is implicitly included in the conveyor system control. They are quite expensive and inflexible and typically used for high-throughput final assembly.

When the variety is larger but still limited, and the production volumes for each product type are still quite large (thus not allowing for an effective and efficient use of product layouts as job shops), asynchronous lines can be adopted. In such lines, the part advancement between the different stations is not synchronized. Stations need buffering capacity to accommodate the resulting WIP, and planning and scheduling techniques are to be used for their management.

In both cases, how to manage the line (by controlling the conveyor system or by adopting scheduling approaches) strictly depends on the performance measure to be

CONTACT E. Pastore. Email: erica.pastore@polito.it

optimized. When respecting due dates is crucial, the minimization of the number of late jobs or job delay should be considered (Allahverdi, Aydilek, & Aydilek, 2016). Instead, in capital intensive systems that need to work at maximum capacity, system management should avoid, or minimize, inactivity periods.

If due dates are not relevant, the most common performance measures are the total completion time and the makespan (Pan & Ruiz, 2013; Reza Hejazi & Saghafian, 2005). Minimizing the total completion time implies letting each job finish as soon as possible, thus improving the service level offered to customers. As it is related to the average waiting time, it is also related to the average WIP, due to Little's law (Little, 1961). Thus, it is usually a crucial performance measure in all the industries in which inventory space is limited and/or holding costs are relevant. Instead, the makespan considers the completion time of the last job of the schedule, hence, it is strictly related to the utilization rate. In fact, the sooner the last job completes, the more resources are kept working, i.e., the smaller is the idle/inactivity time and, hence, the higher the utilization rate (which is the ratio between the working time and the total available time).

Usually, total completion time and makespan are separately considered, and only few examples in the literature address them together, as it will be discussed in the next section. However, in real contexts, focusing on both of them at the same time allows for keeping the WIP (and the related space/cost) under control while not loosing too much from a utilization perspective. This is particularly relevant in all the systems producing high value items with capital intensive resources, such as the automotive and the semiconductor industries.

This article considers both the total completion time and the makespan in flowshops, by addressing the minimization of the total completion time with a constraint on the maximum allowed makespan (i.e., on the minimum allowed utilization). Indeed, reducing the impact of WIP is usually a more sensible performance measure in terms of costs and inefficiency; however, low utilization is generally not appreciated in companies. Both the cases of *no-wait* and *regular* (i.e., systems without the no-wait condition) flowshops are considered. No-wait condition refers to the impossibility for a job to wait between one operation and the next, i.e., once the job is started, all its operations need to be performed one after the other with no waiting time in-between. This situation can be related to space unavailability between each pair of workstation or it can more likely be due to technological reasons (e.g., Aydilek, Aydilek, Allahverdi, and Allahverdi (2022); Na, Ahmed, Nemhauser, and Sokol (2014); Pastore, Alfieri, and Castiglione (2023); Yuan, Jing, Huang, Ren, et al. (2013)). Also, a permutation flowshop is considered, i.e., each machine processes the jobs with the same identical sequence. The problem is mathematically formulated, and heuristics algorithms are designed for its solution and tested on literature available benchmark instances.

The reminder of the article is organized as follows. In section 2, the most relevant literature on scheduling problems related to the one addressed in the article is revised, and the contribution of the article is stated. Section 3, 4 and 5 present the mathematical model, the solution algorithms and discuss the achieved results, respectively. Finally, section 6 concludes the article.

2. Literature Review

The literature on scheduling problems minimizing either the makespan or the total completion time is vast (Aydilek, Aydilek, & Allahverdi, 2021; Pan & Ruiz, 2013;

Reza Hejazi & Saghafian, 2005). Schedules that minimize both the makespan and the total completion time are usually referred to as *ideal schedules* (Coffman, Sethuraman, & Timkovsky, 2003), however they exist only for a subset of problem instances with specific characteristics (Jiang, Lee, & Pinedo, 2021). In the literature restricted to flowshop systems, authors considered these two performance measures together either by (i) optimizing them in a bi-objective environment, or (ii) by optimizing them in a single-objective weighted sum of total completion time $\sum_j c_j$ and makespan c_{max}, or by (iii) using a constrained optimization in which one of them is optimized in the objective function (o.f.) while the value of the other is limited by using a constraint. Given two objective functions A and B, the mathematical representation of the three optimization alternatives are: (A, B) for the bi-objective optimization, $(\alpha A + \beta B)$ for the single-objective weighted sum, and $\mathcal{E}(A|B)$ for the constrained optimization (T'kindt & Billaut, 2006).

The following review of the literature is limited to flowshop systems in which the makespan and the total completion time are considered together, and addressed with one of the three optimization alternatives.

Table 1 reports a summary of the revised articles (rows) and related characteristics (columns). The first two columns identify the reference of the addressed article and its problem according to Graham's notation (Graham, Lawler, Lenstra, & Kan, 1979). Articles dealing with multi-objective functions are identified by a check mark on column *multi-obj.* (with additional *w.s.* meaning that the single-objective weighted sum is considered). For each article, columns c_{max} and $\sum_j c_j$ identify whether the makespan (c_{max}) and the total completion time ($\sum_j c_j$) are in the objective function (*o.f.*) or in a constraint (*Constr*), respectively. The no-wait condition, the use of permutation flowshops, and the inclusion of setup times are identified by columns *nwt*, *prmu* and *setup*, respectively. The last column (*other*) reports additional relevant characteristics of the studied problem. The last two rows of the table summarize the characteristics of the problems studied in this article.

As shown in Table 1, all the articles addressing regular or no-wait flowshop scheduling problems consider the permutation case, as this largely simplifies the structure of the problem. The permutation assumption is not made in Buddala, Mahapatra, and Singh (2022); M. Marichelvam, Geetha, and Tosun (2020); M. K. Marichelvam, Prabaharan, and Yang (2014), as they consider hybrid or flexible flowshops. All of them developed meta-heuristic algorithms to solve the problem. Specifically, M. K. Marichelvam et al. (2014) and M. Marichelvam et al. (2020) considered an hybrid flowshop system with $\alpha c_{max} + \beta \sum_j c_j$ as o.f., while Buddala et al. (2022) addressed a flexible bi-objective flowshop problem.

The flowshop scheduling problem with bi-objective function including the minimization of c_{max} and $\sum_j c_j$ has been mainly addressed by developing heuristic or meta-heuristic algorithms. The majority of the literature addressing this problem have considered regular flowshops, except for Laha and Gupta (2016), which focused on no-wait systems and proposed a Hungarian penalty-based construction heuristic. The special case of the two-machine flowshop system has been studied by Jiang, Lee, and Pinedo (2023), which developed approximation algorithms valid when processing times are proportionate, job ordered or machine ordered. Some of the most recent studies that addressed the case of m machines are: Balasundaram, Valavan, and Baskar (2014); M. Marichelvam, Tosun, and Geetha (2017); Pasupathy, Rajendran, and Suresh (2006); Pugazhenthi and Anthony Xavior (2013); Ravindran, Selvakumar, Sivaraman, and Haq (2005); Zangari, Mendiburu, Santana, and Pozo (2017); Zhao, He, and Liu

(2017). All these authors developed heuristic and meta-heuristic methods to solve the $Fm|prmu|(c_{max}, \sum_j c_j)$ problem. A more comprehensive review of this problem can be found in M. Marichelvam et al. (2017). Other studies included more specific characteristics, among which He, Li, Zhang, and Cao (2019) considered sequence-independent setup times and minimized a multi-objective function that includes: makespan, total completion time, total production cost, and idle time.

In other articles, the two objective functions are combined in a single weighted sum of makespan and total completion time. Among them, Allahverdi and Aldowaisan (2002) and Ye, Li, and Nault (2020) considered no-wait flowshops. While Allahverdi and Aldowaisan (2002) proposed a branch and bound algorithm for the special $F2$ case and some heuristics for the general Fm case, Ye et al. (2020) proposed a trade-off balancing heuristic for the Fm case. Regular flowshops, instead, have been addressed by various authors as, for example, Cheng, Tadikamalla, Shang, and Zhang (2015); Nugraheni, Abednego, and Saputra (2022); Rajkumar and Robert (2019); Sanjeev Kumar, Padmanaban, and Rajkumar (2018); Wang and Zhang (2015). Cheng et al. (2015) considered the special case of $F2$ systems, by finding dominance rules, devising polinomially-solvable cases and proposing a branch and bound algorithm. All the others developed heuristic and meta-heuristic algorithms to solve the problem with a more general Fm system.

Other studies considered constrained optimization problems for addressing two objective functions together in permutation flowshops. Aydilek and Allahverdi (2012); Framinan and Leisten (2006); Nagano, de Almeida, and Miyata (2021) considered the case of minimizing the makespan subject to a total completion time constraint (Aydilek & Allahverdi, 2012; Nagano et al., 2021) or to a maximum tardiness constraint (Framinan & Leisten, 2006). Instead, Allahverdi, Aydilek, and Aydilek (2018, 2020, 2022) considered the problem of minimizing the total tardiness subject to a makespan constraint.

To the authors' knowledge, only four articles considered the minimization of the total completion time subject to a makespan constraint (Allahverdi & Aydilek, 2013, 2014; Almeida & Nagano, 2023; Cheng, Tadikamalla, Shang, & Zhang, 2014). Among these, Allahverdi and Aydilek (2013, 2014); Almeida and Nagano (2023) studied the no-wait case, while Cheng et al. (2014) the regular case. Allahverdi and Aydilek (2013) proposed a heuristic algorithm to set a feasible upper-bound for the makespan constraint, and developed a heuristic algorithm (PAL) based on sequence permutation and a genetic algorithm to solve the $Fm|nwt|\mathcal{E}(\sum_j c_j/c_{max})$ problem. Their results on randomly generated instances showed that PAL algorithm outperforms the genetic one. The same problem with non-zero setup times has been addressed by Allahverdi and Aydilek (2014); Almeida and Nagano (2023), which considered sequence-independent and sequence-dependent setup times, respectively. Allahverdi and Aydilek (2014) developed ad-hoc heuristics, among which insertion based, genetic, simulated annealing and differential evolution algorithms. Instead, Almeida and Nagano (2023) compared the best algorithms found in Allahverdi and Aydilek (2013) and Allahverdi and Aydilek (2014) with four new heuristics based on destruction and construction phases and acceptance criteria. The permutation flowshop system with zero setup times is considered by Cheng et al. (2014), which minimized the total completion time subject to a makespan constraint, and addressed the special case of deteriorating jobs (i.e., job processing times increase with the increase of the starting time). They proposed two algorithms based on pairwise interchange of jobs and a branch and bound algo-

rithm using dominance rules to reduce the dimensions of the search tree. Polynomial time solvable cases are identified and the proposed algorithms are tested on randomly generated instances.

2.1. Contribution

This article addresses the permutation flowshop scheduling problem in both the regular and the no-wait conditions with the aim to minimize the total completion time subject to a makespan constraint. Using Graham's notation (Graham et al., 1979), the addressed problems are: $Fm|prmu|\mathcal{E}(\sum_j c_j/c_{max})$ and $Fm|nwt|\mathcal{E}(\sum_j c_j/c_{max})$.

As shown by Table 1 and discussed in the literature review section, this is the first attempt in the literature of addressing the regular $Fm|prmu|\mathcal{E}(\sum_j c_j/c_{max})$ problem. In fact, Cheng et al. (2014) considered the same problem; however, they limited the study to the case of deteriorating jobs. Instead, the no-wait $Fm|nwt|\mathcal{E}(\sum_j c_j/c_{max})$ problem has been already addressed by Allahverdi and Aydilek (2013); thus, the numerical results will be compared with their results.

Two local search algorithms are developed to solve the problem. The first algorithm considers a sliding-window neighbourhood structure, while the second considers swap neighbourhoods. Both local search algorithms are able to deal with both no-wait and regular systems. They are compared with the methods proposed in Allahverdi and Aydilek (2013), which have also been adapted for the regular flowshop in this article. The algorithms are tested on Taillard's benchmark (Taillard, 1993), and the results show that the proposed local search algorithms are more efficient than the state-of-the-art.

3. Mathematical formulation

The considered system is a flowshop composed by m machines and n jobs. Each job j ($j = 1, \ldots, n$) has a processing time on machine i ($i = 1, \ldots, m$) defined as $p_{j,i} \in \mathbb{R}_+$.

In the following, the MILP formulation of the addressed problem, based on positional variables, is proposed. In the model, M is the set of machine indexes $\{1, \ldots, m\}$, J is the set of job indexes $\{1, \ldots, n\}$, and P is the set of all the possible positions $\{1, \ldots, n\}$ where a job can be scheduled. The maximum makespan value is denoted as K.

The 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 $m \in M$;
- $c_{max} \in \mathbb{R}_+$: makespan.

The MILP is as follows:

Table 1.: Summary of the state of the art.

Article	**Graham notation**	**optimization**	$\mathbf{c_{max}}$	$\sum_j \mathbf{c_j}$	**nwt**	**prmu**	**setup**	**other**
Buddala et al. (2022)	$FFm\|\|c_{max}, \sum_j c_j$	✓	o.f.	o.f.				
M. K. Marichelvam et al. (2014)	$FHm\|\|c_{max}, \sum_j c_j$	✓(w.s.)	o.f.	o.f.				
M. Marichelvam et al. (2020)	$FHm\|hum.f.\|c_{max}, \sum_j c_j$	✓(w.s.)	o.f.	o.f.				human factors
Laha and Gupta (2016)	$Fm\|nwt\|c_{max}, \sum_j c_j$	✓	o.f.	o.f.	✓	✓		
Jiang et al. (2023)	$F2\|prmu\|c_{max}, \sum_j c_j$	✓	o.f.	o.f.		✓		ordered/proportionate jobs
Balasundaram et al. (2014)	$Fm\|prmu\|c_{max}, \sum_j c_j$	✓	o.f.	o.f.		✓		
M. Marichelvam et al. (2017)	$Fm\|nwt\|c_{max}, \sum_j c_j$	✓	o.f.	o.f.		✓		
Pasupathy et al. (2006)	$Fm\|nwt\|c_{max}, \sum_j c_j$	✓	o.f.	o.f.		✓		
Pugazhenthi and Anthony Xavior (2013)	$Fm\|prmu\|C_{max}, \sum_j c_j$	✓	o.f.	o.f.		✓		
Ravindran et al. (2005)	$Fm\|prmu\|c_{max}, \sum_j c_j$	✓	o.f.	o.f.		✓		
Zangari et al. (2017)	$Fm\|nwt\|c_{max}, \sum_j c_j$	✓	o.f.	o.f.		✓		
Zhao et al. (2017)	$Fm\|nwt\|c_{max}, \sum_j c_j$	✓	o.f.	o.f.		✓		
He et al. (2019)	$Fm\|prmu, s_{ijk}\|c_{max}, \sum_j c_j, IT, prod.cost$	✓	o.f.	o.f.		✓	✓	
Allahverdi and Aldowaisan (2002)	$Fm\|nwt\|c_{max}, \sum_j c_j$	✓(w.s.)	o.f.	o.f.	✓	✓		
Ye et al. (2020)	$Fm\|nwt\|c_{max}, \sum_j c_j$	✓(w.s.)	o.f.	o.f.	✓	✓		
Nugraheni et al. (2022)	$Fm\|prmu\|c_{max}, \sum_j c_j$	✓(w.s.)	o.f.	o.f.		✓		
Rajkumar and Robert (2019)	$Fm\|prmu\|c_{max}, \sum_j c_j$	✓(w.s.)	o.f.	o.f.		✓		
Sanjeev Kumar et al. (2018)	$Fm\|prmu\|c_{max}, \sum_j c_j$	✓(w.s.)	o.f.	o.f.		✓		
Cheng et al. (2015)	$F2\|prmu, det.\|c_{max}, \sum_j c_j$	✓(w.s.)	o.f.	o.f.		✓		deteriorating jobs
Wang and Zhang (2015)	$Fm\|prmu\|c_{max}, \sum_j c_j$	✓(w.s.)	o.f.	o.f.		✓		learning effect
Nagano et al. (2021)	$Fm\|nwt\|\mathcal{E}(c_{max}/\sum_j c_j)$		o.f.	Constr	✓	✓		
Aydilek and Allahverdi (2012)	$Fm\|nwt\|\mathcal{E}(c_{max}/\sum_j c_j)$		o.f.	Constr	✓	✓		
Framinan and Leisten (2006)	$Fm\|prmu\|\mathcal{E}(c_{max}/T_{max})$		o.f.			✓		Constr. on maximum tardiness
Allahverdi et al. (2018)	$Fm\|nwt\|\mathcal{E}(\sum_j T_j/c_{max})$		Constr		✓	✓		min. total tardiness
Allahverdi et al. (2020)	$Fm\|nwt, s_{ij}\|\mathcal{E}(\sum_j T_j/c_{max})$		Constr		✓	✓	✓	min. total tardiness
Allahverdi et al. (2022)	$Fm\|nwt\|\mathcal{E}(\sum_j T_j/\sum_j c_j)$			Constr	✓	✓		min. total tardiness
Allahverdi and Aydilek (2014)	$Fm\|nwt, s_{ij}\|\mathcal{E}(\sum_j C_j/c_{max})$		Constr	o.f.	✓	✓	✓	
Almeida and Nagano (2023)	$Fm\|nwt, s_{ijk}\|\mathcal{E}(\sum_j c_j/c_{max})$		Constr	o.f.	✓	✓	✓	sequence-dependent setups
Allahverdi and Aydilek (2013)	$Fm\|nwt\|\mathcal{E}(\sum_j c_j/c_{max})$		Constr	o.f.	✓	✓		
Cheng et al. (2014)	$Fm\|prmu, det.\|\mathcal{E}(\sum_j c_j/c_{max})$		Constr	o.f.		✓		deteriorating jobs
This article	$Fm\|prmu\|\mathcal{E}(\sum_j c_j/c_{max})$		Constr	o.f.		✓		
	$Fm\|nwt\|\mathcal{E}(\sum_j c_j/c_{max})$		Constr	o.f.	✓	✓		

$$\min \quad \sum_{k \in P} c_{k,m} \tag{1}$$

$$\text{s.t.} \quad \sum_{j \in J} x_{j,k} = 1 \qquad \forall k \in P \tag{2}$$

$$\sum_{k \in P} x_{j,k} = 1 \qquad \forall j \in J \tag{3}$$

$$c_{1,1} = \sum_{j \in J} x_{j,1} p_{j,1} \tag{4}$$

$$c_{k,i} \geq c_{k,i-1} + \sum_{j \in J} x_{j,k} p_{j,i} \qquad \forall k \in P, i \in M - \{1\} \tag{5}$$

$$c_{k,i} \geq c_{k-1,i} + \sum_{j \in J} x_{j,k} p_{j,i} \qquad \forall k \in P - \{1\}, i \in M \tag{6}$$

$$c_{max} = c_{n,m} \tag{7}$$

$$c_{max} \leq K \tag{8}$$

$$x_{j,k} \in \{0,1\} \qquad \forall j \in J, k \in P \tag{9}$$

$$c_{k,i} \in \mathbb{R}_+ \qquad \forall i \in M, k \in P \tag{10}$$

Constraints (2) and (3) state that each position is assigned to a single job, and each job is assigned to one position, respectively. Constraint (4) forces the job in the first schedule position to start at time zero on the first machine. Constraints (5) and (6) are the flowshop timing constraints, establishing the completion time of each job on each machine. Constraint (8) imposes an upper bound K for the makespan, which is computed in (7). Finally, (9) and (10) define the domain of the decision variables.

To consider no-wait schedules, constraints (5) become:

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

4. Solution algorithms

Two local search algorithms are developed for the problem introduced in the previous sections, i.e., the minimization of the total completion time with a constraint on the makespan in permutation flowshops. The algorithms are based on the swap and on the sliding window neighbourhood structures, respectively. Both are fed with the same initial solution, which is found by a NEH-inspired heuristic. The two proposed algorithms can both be used for regular and no-wait conditions, by adapting the way the makespan and the total completion time of a schedule are computed. For each incumbent solution π, the makespan and the total completion time are numerically calculated both for the regular and for the no-wait permutation flowshops.

The upper bound value K for the makespan constraint is found with the *K-algorithm* (KA) proposed in Allahverdi and Aydilek (2013) (which has also been adapted, in this article, for the regular flowshop case). The solution found by the K-algorithm is called π_{KA} in the following.

4.1. Initial solution

The initial solution π is generated by a NEH-based heuristic, called NBH in the article. Such heuristic minimizes the weighted sum of the makespan (c_{max}) and the total completion time (c_{tot}). Its objective function is as follows:

$$OF_{NBH} = \alpha\ c_{max} + (1 - \alpha)c_{tot}. \tag{12}$$

Algorithm 1 NEH-based Heuristic algorithm (NBH)

Input: K, π_{KA}
Output: π_{NBH}

1: $\pi_{NBH} \leftarrow \pi_{KA}$;
2: $c^*_{tot} \leftarrow$ total completion time of of π_{KA};
3: **for all** $a \in \{0.8, 0.6, 0.4, 0.2\}$ **do**
4: $\quad\alpha \leftarrow a$;
5: $\quad\pi \leftarrow NEH(a)$;
6: $\quad c_{max,\pi} \leftarrow$ makespan of solution π;
7: $\quad c_{tot,\pi} \leftarrow$ total completion time of of π;
8: $\quad$**if** $c_{max,\pi} \leq K$ and $c_{tot,\pi} < c^*_{tot}$ **then**
9: $\quad\quad c^*_{tot} \leftarrow c_{tot,\pi}$
10: $\quad\quad \pi_{NBH} \leftarrow \pi$
11: $\quad$**end if**
12: **end for**

Th pseudo-code of NBH is reported in Algorithm 1. It starts from the initial solution found by the K-Algorithm (π_{KA}). For various values of the parameter α, i.e., the various weights of the objective function in equation (12), the NEH algorithm (Nawaz, Enscore Jr, & Ham, 1983) is used to find an incumbent solution π. The final solution π_{NBH} is the best feasible solution among the found incumbents. The structure of the NEH algorithm is the standard one by Nawaz et al. (1983); however, the evaluated objective function at each iteration is the one reported in equation (12).

4.2. Sliding window local search

One of the key elements in a local search algorithm is the definition of the neighbourhood, i.e., of the set of solutions close to the incumbent to be analyzed to look for possible improvements.

In the sliding window local search algorithm, the neighbourhood $\mathcal{N}^1_{\pi,\psi,w_P}$ of an incumbent solution π is defined by an initial position ψ and a size parameter w_P. The neighbourhood is composed of all the solutions that can be obtained by fixing all the positions of π not included in the *job-window* $\pi(w_P, \psi)$, where $\pi(w_P, \psi)$ is the index set of jobs located in the consecutive positions $\psi, \psi + 1, \ldots, \psi + w_P - 1$ of π. As an example, given a problem with $n = 8$ jobs, the neighbourhood $\mathcal{N}^1_{\pi,4,3}$ of the solution $\pi = [j_1, j_2, j_3, j_4, j_5, j_6, j_7, j_8]$ starting at position $\psi = 4$ and of size $w_p = 3$ is shown in Table 2. In the table, the *job-window* is highlighted in bold.

The pseudo-code of the sliding window local search (SW) is devised in Algorithm 2. The algorithm takes as input: the initial solution π_{NBH} of the NBH algorithm, the window size w_P and a time limit θ. It gives as output the heuristic solution π^*. At each iteration of the algorithm, the best solution in $\mathcal{N}^1_{\pi,\psi,w_P}$ is found by using the `MILP` (line 5 of Algorithm 2), and stored if it is both feasible and better than the best

Algorithm 2 Sliding window local search (SW)

Input: π_{NBH}, K, θ
Output: π^*

1: $\psi \leftarrow 0$;
2: $\pi \leftarrow \pi_{NBH}$
3: $c^*_{tot} \leftarrow ctot_{\pi_{NBH}}$;
4: **while** θ is not reached **do**
5: $\quad \pi \leftarrow$ best solution in $\mathcal{N}^1_{\pi,\psi,w_P}$ computed by means of `MILP`
6: $\quad c_{max,\pi} \leftarrow$ makespan of solution π;
7: $\quad c_{tot,\pi} \leftarrow$ total completion time of π;
8: $\quad$ **if** $c_{max,\pi} \leq K$ and $c_{tot,\pi} < c^*_{tot}$ **then**
9: $\quad\quad c^*_{tot} \leftarrow c_{tot,\pi}$;
10: $\quad\quad \pi^* \leftarrow \pi$;
11: $\quad$ **end if**
12: $\quad \psi \leftarrow \psi + 1$;
13: $\quad$ **if** $\psi > n - w_P$ **then**
14: $\quad\quad \psi \leftarrow 0$
15: $\quad$ **end if**
16: **end while**

$[j_1, j_2, j_3, j_4, \mathbf{j_6}, \mathbf{j_5}, \mathbf{j_7}, j_8]$	$[j_1, j_2, j_3, j_4, \mathbf{j_7}, \mathbf{j_6}, \mathbf{j_5}, j_8]$	$[j_1, j_2, j_3, j_4, \mathbf{j_6}, \mathbf{j_7}, \mathbf{j_5}, j_8]$
$[j_1, j_2, j_3, j_4, \mathbf{j_5}, \mathbf{j_7}, \mathbf{j_6}, j_8]$	$[j_1, j_2, j_3, j_4, \mathbf{j_7}, \mathbf{j_5}, \mathbf{j_6}, j_8]$	

Table 2.: Sliding window neighbourhood $\mathcal{N}^1_{\pi,4,3}$ of the solution $\pi = [j_1, j_2, j_3, j_4, j_5, j_6, j_7, j_8]$.

solution found so far, i.e., π^* (lines 8-11). At each iteration, the job-window *slides*, as the first job position ψ increases by one unit (line 12). The algorithm stops when the time limit θ is reached.

In Algorithm 2, incumbents solutions are found by means of `MILP` (line 5). At each iteration, `MILP` is solved to optimality by a commercial solver; in the `MILP` , the values of schedule positions outside the job-window are fixed to the values of the best solution, while the positions inside the job-window are optimized by the solver.

4.3. Swap local search

The local search algorithm based on the swap neighbourhoods is similar to that using the sliding windows and previously described. The swap neighbourhood $\mathcal{N}^2_\pi$ of a candidate solution π is given by all the possible swaps between positions k and l (with $l > k$) in π. As an example, given a problem with $n = 4$ jobs, the swap neighbourhood $\mathcal{N}^2_\pi$ of solution $\pi = [j_1, j_2, j_3, j_4]$ is given in Table 3. In the table, for each neighbour, the swap is highlighted in bold. To give an example, in the fist neighbour $[\mathbf{j_2}, \mathbf{j_1}, j_3, j_4]$ of Table 3 the first two jobs of the schedule (j_1 and j_2) are swapped (i.e., j_1 moves from position 1 to position 2 of the schedule, and j_2 moves from position 2 to position 1 of the schedule).

$[\mathbf{j_2}, \mathbf{j_1}, j_3, j_4]$	$[\mathbf{j_3}, j_2, \mathbf{j_1}, j_4]$	$[\mathbf{j_4}, j_2, j_3, \mathbf{j_1}]$
$[j_1, \mathbf{j_3}, \mathbf{j_2}, j_4]$	$[j_1, \mathbf{j_4}, j_3, \mathbf{j_2}]$	$[j_1, j_2, \mathbf{j_4}, \mathbf{j_3}]$

Table 3.: Swap neighbourhood $\mathcal{N}^2_\pi$ of the solution $\pi = [j_1, j_2, j_3, j_4]$.

The swap local search mechanism works similarly to SW, and its pseudo-code is shown in Algorithm 3. At each iteration, the best solution is found by exploring $\mathcal{N}_\pi^2$ (line 4 of Algorithm 3). Differently from SW, the `MILP` is not used to explore the neighbourhood as it can be numerically found by modifying the sequence of π. Candidate solutions are evaluated in terms of feasibility ($c_{max,\pi} < K$) and in terms of objective function ($c_{tot,\pi}$). When the stopping criterion (i.e., the θ time limit) is reached, the best found solution π^* is given as output of the algorithm.

Algorithm 3 Swap local search (SWAP)

Input: π_{NBH}, K, θ
Output: π^*

1: $\pi \leftarrow \pi_{NBH}$
2: $c^*_{tot} \leftarrow ctot_{\pi_{NBH}}$;
3: **while** θ is not reached **do**
4: $\quad \pi \leftarrow$ best solution in $\mathcal{N}^2_{\pi,\psi,w_P}$
5: $\quad c_{max,\pi} \leftarrow$ makespan of solution π;
6: $\quad c_{tot,\pi} \leftarrow$ total completion time of π;
7: $\quad$ **if** $c_{max,\pi} \leq K$ and $c_{tot,\pi} < c^*_{tot}$ **then**
8: $\quad\quad c^*_{tot} \leftarrow c_{tot,\pi}$;
9: $\quad\quad \pi^* \leftarrow \pi$;
10: $\quad$ **end if**
11: **end while**

5. Numerical results

5.1. Design of experiment

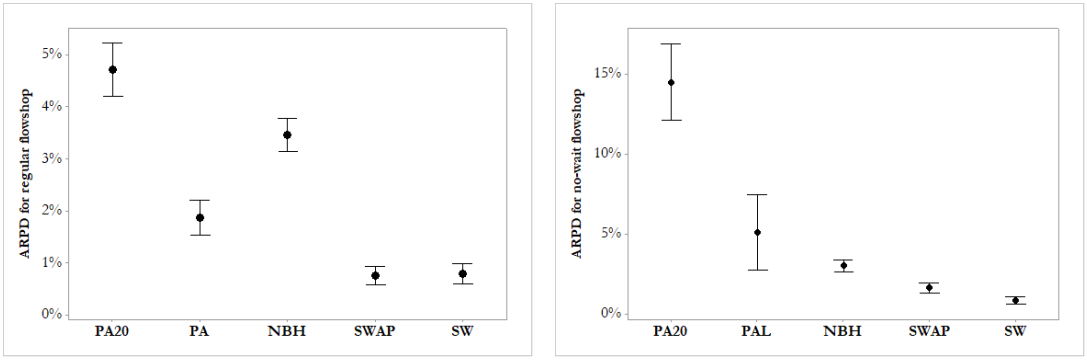
The proposed algorithms are tested on the Taillard benchmark set of instances (Taillard, 1993). This set is composed of 12 problems with different number of jobs n and machines m. The number of jobs n varies between 20 and 500, while m varies between 5 and 20. Ten instances are available for each problem, thus the benchmark set contains a total number of 120 instances.

In the experiment, the proposed swap and sliding window local search algorithms are compared with PAL algorithm by Allahverdi and Aydilek (2013). Instead, the K-algorithm by Allahverdi and Aydilek (2013) has been used to set the makespan upperbound K. Both the algorithms by Allahverdi and Aydilek (2013), designed for the no-wait problem, have been adapted for the regular flowshop by changing the way the objective function is numerically computed.

Each algorithm has been run with a time limit θ depending on the problem size, computed as $\theta = \frac{60nm}{1000}$ (Alfieri, Garraffa, Pastore, & Salassa, 2023; Balogh, Garraffa, O'Sullivan, & Salassa, 2022), and measured in seconds. Although in Allahverdi and Aydilek (2013) PAL algorithm is run with a fixed number of iterations L, in this article, it has been implemented by letting L vary and by fixing the time limit θ. This makes the comparison among all the algorithms fair from a computation time standpoint.

The parameter window size (w_P) of the sliding window algorithm has been set to 8 jobs, after preliminary tests, to assure that, on average, the time to explore the neighbourhood is less than 10 seconds.

All the algorithms have been implemented in Java, and CPLEX 12.10 have been used as MILP solver. Tests have been run on a Intel(R) Core(TM) i7-8700K CPU processor at 3.70 GHz, with 32 GB of RAM.



(a) Regular flowshop.

(b) No-wait flowshop.

Figure 1.: ARPD 95% condifence intervals of the tested algorithms, grouped by regular/no-wait condition.

5.2. Results

As all the tested algorithms are run with the same time limit, the comparison is performed on the quality of the solutions.
For the comparison, the average relative percentage deviation ARPD is used. Specifically, for each instance of the benchmark set, the RPD of algorithm k is computed as

$$RPD_k = 100 \times \frac{OF_k - OF^*}{OF^*}, \tag{13}$$

where $OF^* = \min_{k \in \mathcal{K}} OF_k$, i.e., OF^* is the best solution in the set $\mathcal{K}$ of the tested algorithms. The set of algorithms $\mathcal{K}$ includes: the sliding window local search algorithm (SW), the swap local search algorithm (SWAP), the NEH-based heuristic (NBH), the PAL algorithm (PAL) (with the fixed time limit and a varying number of iteration L), and the K-algorithm (KA) by Allahverdi and Aydilek (2013). As Allahverdi and Aydilek (2013) proved that the best performance for PAL is achieved by setting $L = 20$ (i.e., by setting a fixed number of iterations equal to 20), Table 4 and Figure 1 also includes the results of PA20 for reference, even though its computation time is largely shorter than the fixed time limit used for all the other algorithms.

Table 4 shows the ARPD percentage of the tested algorithms for the regular and the no-wait flowshops, for each combination of number of jobs n and number of machines m. For each row, the algorithm with the best performance, i.e., that with the lowest ARPD value, is highlighted in bold. For each flowshop case (i.e., regular and no-wait), the two last rows display the ARPD among all the instances (row **total**) and the percentage of instances in which the algorithm found the best solution (row **% best**). The percentages of such row do not sum to 100% as there are instances in which more than one algorithm find the best solution. Figure 1, instead, shows the 95% confidence intervals of the tested algorithms. The graphs are divided for the regular and the no-wait cases; the KA algorithm is not shown because its ARPD values and confidence intervals are much larger than the other algorithms, as its average values reported in Table 4 show.

For the regular flowshop, the proposed local search algorithms (SWAP and SW) perform better than the others. The results are confirmed by the 95% confidence interval plot in Figure 1(a), from which SWAP and SW are shown to be statistically

Regular flowshop

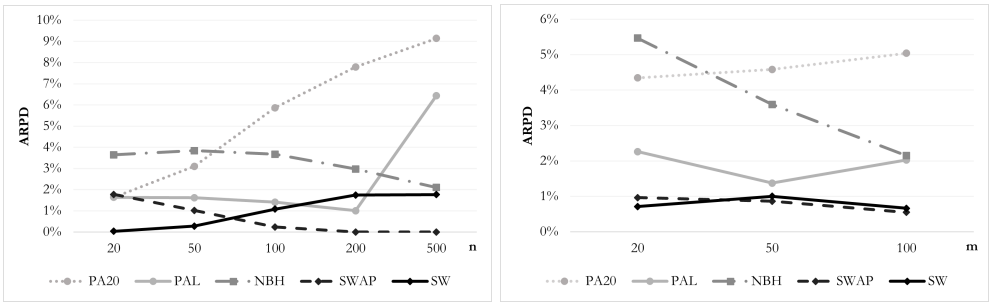
n	m	KA	PA20	PAL	NBH	SWAP	SW
20	5	23.47%	1.51%	1.51%	4.45%	1.93%	**0.04%**
	10	18.42%	1.82%	1.82%	3.51%	2.03%	**0.00%**
	20	13.86%	1.60%	1.60%	2.98%	1.36%	**0.07%**
50	5	27.59%	4.57%	3.05%	5.64%	0.96%	**0.26%**
	10	21.87%	2.49%	0.81%	3.40%	1.15%	**0.21%**
	20	18.65%	2.24%	0.99%	2.48%	0.96%	**0.37%**
100	5	24.24%	6.96%	2.24%	6.31%	**0.00%**	1.84%
	10	20.74%	5.58%	1.53%	3.13%	**0.27%**	0.81%
	20	18.75%	5.07%	0.45%	1.58%	**0.43%**	0.60%
200	10	20.34%	8.44%	1.34%	4.35%	**0.00%**	2.98%
	20	18.17%	7.15%	0.67%	1.59%	**0.00%**	0.51%
500	20	15.79%	9.15%	6.44%	2.10%	**0.00%**	1.77%
total		20.16%	4.71%	1.87%	3.46%	**0.76%**	0.79%
% best		0%	2%	10%	0%	44%	46%

No-wait flowshop

n	m	KA	PA20	PAL	NBH	SWAP	SW
20	5	30.74%	1.36%	1.36%	4.56%	3.35%	**0.26%**
	10	28.98%	1.94%	1.94%	4.57%	3.39%	**0.29%**
	20	22.58%	**0.36%**	**0.36%**	2.63%	1.79%	**0.36%**
50	5	42.81%	5.61%	1.52%	5.01%	2.33%	**1.44%**
	10	43.43%	6.01%	**0.59%**	3.48%	2.28%	1.35%
	20	41.26%	6.28%	**0.81%**	2.86%	1.70%	1.11%
100	5	49.65%	17.18%	0.40%	2.85%	**0.31%**	0.62%
	10	51.06%	18.12%	**0.00%**	4.73%	2.66%	2.92%
	20	47.92%	16.27%	**0.26%**	1.98%	1.48%	0.90%
200	10	55.77%	27.99%	2.36%	1.78%	**0.03%**	0.71%
	20	55.25%	28.00%	4.15%	1.03%	0.21%	**0.15%**
500	20	67.32%	44.80%	47.61%	0.72%	**0.06%**	0.14%
total		44.73%	14.49%	5.11%	3.02%	1.63%	**0.85%**
% best		0%	8%	38%	0%	21%	42%

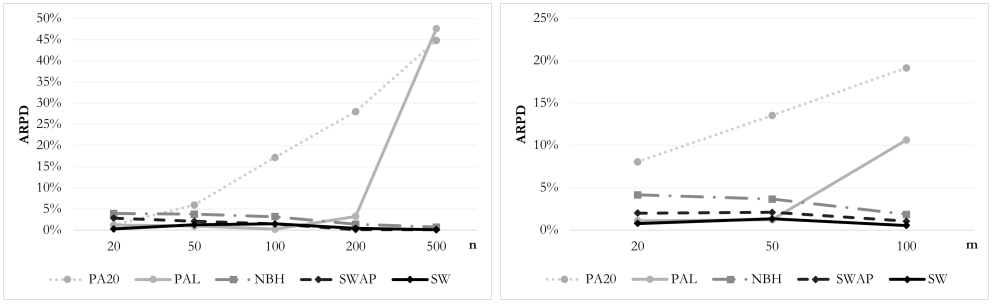
Table 4.: ARPD values, grouped by regular/no-wait condition, and by n and m factor levels.

different and less from the other algorithms. Also, there is no statistical difference between SWAP and SW: a Mann-Whitney test confirmed that the medians of the two algorithms are statistically equal with a confidence of 97.33 %. Such non-parametric test is used because the RPD values may not follow a Normal distribution, as they cannot assume negative values (Costa, Pastore, & Frigerio, 2021). Table 4 shows that when the problem size increases (i.e., for larger values of n and m), the SWAP algorithm tends to perform better than all the other algorithms while, if the problem size is small, the SW achieves smaller RPD values. Interestingly, while SW is affected by the increasing problem size, SWAP improves its performance. Such behaviour can be appreciated in Figure 2, which shows how the ARPD values of the tested algorithms vary with different values of n and m for the regular flowshop. The same happens when comparing PA20, which worsens for larger values of n, and NBH, which improves (in



(a) ARPD grouped by n levels. (b) ARPD grouped by m levels.

Figure 2.: Regular flowshop ARPD values.



(a) ARPD grouped by n levels. (b) ARPD grouped by m levels.

Figure 3.: No-wait flowshop ARPD values.

Figure 2, the PA20 line increases with n, while NBH decreases). Instead, PAL performs similarly for increasing values of n, but it actually improves with increasing values of m; it drastically worsens for the most larger problems with 500 jobs and 20 machines.

For the no-wait case, Figure 1(b) shows that SWAP and SW perform better than PAL; in this case, the difference between the medians of SWAP and SW is statistically significant, with SW having the lowest RPD values. Table 4 and Figure 3 show that the tested algorithms perform differently according to the number of jobs n and of machines m also for the no-wait case. Both PA20 and PAL worsen their performance when the problem size increases, while NBH and SWAP improve with larger problems. SW, instead, does not show a monotonic behaviour when the problem size grows. As for the regular case, the most difficult instances are poorly solved by both PAL and PA20. This could be related to the initial solutions (i.e., the solutions found by KA) that are largely less efficient than the initial solutions of SWAP and SW (i.e., the solutions found by NBH), and to the incapability of PA20 and PAL of escaping local optima.

For both regular and no-wait flowshops, Table 5 shows the completion time percentage improvement of the proposed algorithms (NBH, SWAP, SW) compared to the state-of-the-art PAL. For each proposed algorithm k, the total completion time percentage improvement Δ_k is computed as:

$$\Delta_k = 100 \times \frac{OF_{PAL} - OF_k}{OF_{PAL}}. \tag{14}$$

		Regular			No-wait		
n	**m**	**NBH**	**SWAP**	**SW**	**NBH**	**SWAP**	**SW**
20	5	-2.91%	-0.43%	1.43%	-3.17%	-1.97%	1.07%
	10	-1.66%	-0.21%	1.78%	-2.59%	-1.44%	1.60%
	20	-1.39%	0.21%	1.49%	-2.27%	-1.43%	-0.01%
50	5	-2.54%	2.01%	2.68%	-3.48%	-0.83%	0.02%
	10	-2.57%	-0.34%	0.59%	-2.90%	-1.70%	-0.78%
	20	-1.49%	0.03%	0.61%	-2.06%	-0.91%	-0.31%
100	5	-4.00%	2.18%	0.38%	-2.44%	0.09%	-0.22%
	10	-1.59%	1.22%	0.70%	-4.73%	-2.66%	-2.92%
	20	-1.13%	0.02%	-0.15%	-1.72%	-1.22%	-0.65%
200	10	-2.97%	1.32%	-1.62%	0.56%	2.27%	1.60%
	20	-0.92%	0.66%	0.15%	2.99%	3.77%	3.83%
500	20	4.08%	6.05%	4.39%	31.77%	32.21%	32.16%

Table 5.: Total completion time percentage improvement of NBH, SWAP and SW compared to PAL, grouped by regular/no-wait condition, and by n and m factor levels.

In Table 5, a positive improvement for algorithm k implies that the total completion time of the solution found by k is lower than that found by PAL, and vice-versa for the negative case. The percentages of Table 5 confirms the results of Table 4 and Figures 1. 2 and 3.

6. Conclusion

Production systems need to optimize both the total completion time and the makespan at the shop floor level to achieve competitiveness. In fact, while the total completion time is related to the waiting time and, hence, to an improvement of the service level offered to customers and of the delivery lead time, a makespan reduction implies a higher utilization of machines/resources. In this article, both these measures are considered in a permutation flowshop scheduling problem. Specifically, the addressed problem minimizes the total completion time with a constraint on the maximum allowed makespan. Both regular and no-wait flowshops are considered. Two local search algorithms are proposed, one based on the swap and the other on the sliding window neighbourhood structures.

The proposed algorithms have been tested and compared with the state-of-the-art heuristics, and the results showed that both swap and sliding window perform better than the state-of-the-art algorithms, for both the regular and the no-wait cases. For the regular flowshop, the two proposed algorithms perform similarly; however, the swap local search tends to achieve better results when the problem size increases. For the no-wait case, instead, the sliding window outperforms the swap local search; however, when the problem becomes larger, the swap guarantees the best completion time values.

Future research will be devoted to a joint optimization of the completion time and the makespan, with the aim of exploring scheduling solutions that are effective from both perspectives.

Disclosure statement

The authors have no conflicts of interest to declare.

Data availability

The data that support the findings of this study are available from the corresponding author, E.P., upon reasonable request.

References

Alfieri, A., Garraffa, M., Pastore, E., & Salassa, F. (2023). Permutation flowshop problems minimizing core waiting time and core idle time. *Computers & Industrial Engineering*, *176*, 108983.

Allahverdi, A., & Aldowaisan, T. (2002). No-wait flowshops with bicriteria of makespan and total completion time. *Journal of the Operational Research Society*, *53*(9), 1004 – 1015.

Allahverdi, A., Aydilek, A., & Aydilek, H. (2016). Minimizing the number of tardy jobs on a two-stage assembly flowshop. *Journal of Industrial and Production Engineering*, *33*(6), 391–403.

Allahverdi, A., & Aydilek, H. (2013). Algorithms for no-wait flowshops with total completion time subject to makespan. *The International Journal of Advanced Manufacturing Technology*, *68*(9-12), 2237 – 2251.

Allahverdi, A., & Aydilek, H. (2014). Total completion time with makespan constraint in no-wait flowshops with setup times. *European Journal of Operational Research*, *238*(3), 724 – 734.

Allahverdi, A., Aydilek, H., & Aydilek, A. (2018). No-wait flowshop scheduling problem with two criteria; total tardiness and makespan. *European Journal of Operational Research*, *269*(2), 590–601.

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

Allahverdi, A., Aydilek, H., & Aydilek, A. (2022). An algorithm for a no-wait flowshop scheduling problem for minimizing total tardiness with a constraint on total completion time. *International Journal of Industrial Engineering Computations*, *13*(1), 43–50.

Almeida, F. S. d., & Nagano, M. S. (2023). Heuristics to optimize total completion time subject to makespan in no-wait flow shops with sequence-dependent setup times. *Journal of the Operational Research Society*, *74*(1), 362 – 373.

Aydilek, H., & Allahverdi, A. (2012). Heuristics for no-wait flowshops with makespan subject to mean completion time. *Applied Mathematics and Computation*, *219*(1), 351–359.

Aydilek, H., Aydilek, A., & Allahverdi, A. (2021). Algorithms to minimize total completion time in a two-machine flowshop problem with uncertain set-up times. *Engineering Optimization*, *53*(8), 1417–1430.

Aydilek, H., Aydilek, A., Allahverdi, M., & Allahverdi, A. (2022). More effective heuristics for a two-machine no-wait flowshop to minimize maximum lateness. *International Journal of Industrial Engineering Computations*, *13*(4), 543–556.

Balasundaram, R., Valavan, D., & Baskar, N. (2014). Heuristic based approach for bi-criteria optimization of minimizing makespan and total flow time of flowshop scheduling. *International Journal of Mechanical & Mechatronics Engineering IJMME-IJENS*, *14*(02).

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.

Buddala, R., Mahapatra, S. S., & Singh, M. R. (2022). Solving multi-objective flexible flow-shop scheduling problem using teaching-learning-based optimisation embedded with maximum deviation theory. *International Journal of Industrial and Systems Engineering*, *42*(1), 39 – 63.

Cheng, M., Tadikamalla, P. R., Shang, J., & Zhang, B. (2015). Two-machine flow shop scheduling with deteriorating jobs: minimizing the weighted sum of makespan and total completion time. *Journal of the Operational Research Society*, *66*(5), 709–719.

Cheng, M., Tadikamalla, P. R., Shang, J., & Zhang, S. (2014). Bicriteria hierarchical optimization of two-machine flow shop scheduling problem with time-dependent deteriorating jobs. *European Journal of Operational Research*, *234*(3), 650 – 657.

Coffman, E., Sethuraman, J., & Timkovsky, V. G. (2003). Ideal preemptive schedules on two processors. *Acta Informatica*, *39*, 597–612.

Costa, A., Pastore, E., & Frigerio, N. (2021). The server allocation problem with non-identical machines: A meta-heuristic approach. *Computers & Industrial Engineering*, *162*, 107687.

Framinan, J. M., & Leisten, R. (2006). A heuristic for scheduling a permutation flowshop with makespan objective subject to maximum tardiness. *International Journal of Production Economics*, *99*(1-2), 28–40.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. In *Annals of discrete mathematics* (Vol. 5, pp. 287–326). Elsevier.

He, L., Li, W., Zhang, Y., & Cao, Y. (2019). A discrete multi-objective fireworks algorithm for flowshop scheduling with sequence-dependent setup times. *Swarm and Evolutionary Computation*, *51*.

Jiang, X., Lee, K., & Pinedo, M. L. (2021). Ideal schedules in parallel machine settings. *European Journal of Operational Research*, *290*(2), 422–434.

Jiang, X., Lee, K., & Pinedo, M. L. (2023). Bicriteria two-machine flowshop scheduling: approximation algorithms and their limits. *Journal of Scheduling*, 1–26.

Laha, D., & Gupta, J. N. (2016). A hungarian penalty-based construction algorithm to minimize makespan and total flow time in no-wait flow shops. *Computers & Industrial Engineering*, *98*, 373–383.

Little, J. D. (1961). A proof for the queuing formula: L= λ w. *Operations Research*, *9*(3), 383–387.

Marichelvam, M., Geetha, M., & Tosun, (2020). An improved particle swarm optimization algorithm to solve hybrid flowshop scheduling problems with the effect of human factors – a case study. *Computers & Operations Research*, *114*.

Marichelvam, M., Tosun, Ö., & Geetha, M. (2017). Hybrid monkey search algorithm for flow shop scheduling problem under makespan and total flow time. *Applied Soft Computing*, *55*, 82–92.

Marichelvam, M. K., Prabaharan, T., & Yang, X. S. (2014). A discrete firefly algorithm for the multi-objective hybrid flowshop scheduling problems. *IEEE Transactions on Evolutionary Computation*, *18*(2), 301 – 305.

Na, B., Ahmed, S., Nemhauser, G., & Sokol, J. (2014). A cutting and scheduling problem in float glass manufacturing. *Journal of Scheduling*, *17*, 95–107.

Nagano, M. S., de Almeida, F. S., & Miyata, H. H. (2021). An iterated greedy algorithm for the no-wait flowshop scheduling problem to minimize makespan subject to total completion time. *Engineering Optimization*, *53*(8), 1431 – 1449.

Nawaz, M., Enscore Jr, E. E., & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, *11*(1), 91–95.

Nugraheni, C. E., Abednego, L., & Saputra, C. S. (2022). Whale optimization algorithms for multi-objective flowshop scheduling problems. In *Eurasia proceedings of science, technology, engineering and mathematics* (Vol. 21, p. 441 – 451).

Pan, Q.-K., & Ruiz, R. (2013). A comprehensive review and evaluation of permutation flowshop heuristics to minimize flowtime. *Computers & Operations Research*, *40*(1), 117–128.

Pastore, E., Alfieri, A., & Castiglione, C. (2023). Addressing idle and waiting time in short

term production planning. *Materials Research Proceedings*, *35*.

Pasupathy, T., Rajendran, C., & Suresh, R. (2006). A multi-objective genetic algorithm for scheduling in flow shops to minimize the makespan and total flow time of jobs. *The International Journal of Advanced Manufacturing Technology*, *27*, 804–815.

Pugazhenthi, R., & Anthony Xavior, M. (2013). Optimisation of permutation flow shop with multi objective criteria. *International Journal of Applied Engineering Research*, *8*(15), 1807–1813.

Rajkumar, R., & Robert, R. J. (2019). A hybrid algorithm for multi-objective optimization of minimizing makespan and total flow time in permutation flow shop scheduling problems. *Information Technology and Control*, *48*(1), 47–57.

Ravindran, D., Selvakumar, S., Sivaraman, R., & Haq, A. N. (2005). Flow shop scheduling with multiple objective of minimizing makespan and total flow time. *The International Journal of Advanced Manufacturing Technology*, *25*, 1007–1012.

Reza Hejazi, S., & Saghafian, S. (2005). Flowshop-scheduling problems with makespan criterion: a review. *International Journal of Production Research*, *43*(14), 2895–2929.

Sanjeev Kumar, R., Padmanaban, K., & Rajkumar, M. (2018). Minimizing makespan and total flow time in permutation flow shop scheduling problems using modified gravitational emulation local search algorithm. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, *232*(3), 534–545.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, *64*(2), 278–285.

T'kindt, V., & Billaut, J.-C. (2006). *Multicriteria scheduling: theory, models and algorithms.* Springer Science & Business Media.

Wang, J.-J., & Zhang, B.-H. (2015). Permutation flowshop problems with bi-criterion makespan and total completion time objective and position-weighted learning effects. *Computers & Operations Research*, *58*, 24 – 31.

Ye, H., Li, W., & Nault, B. R. (2020). Trade-off balancing between maximum and total completion times for no-wait flow shop production. *International Journal of Production Research*, *58*(11), 3235–3251.

Yuan, H., Jing, Y., Huang, J., Ren, T., et al. (2013). Optimal research and numerical simulation for scheduling no-wait flow shop in steel production. *Journal of Applied Mathematics*, *2013*.

Zangari, M., Mendiburu, A., Santana, R., & Pozo, A. (2017). Multiobjective decomposition-based mallows models estimation of distribution algorithm. a case of study for permutation flowshop scheduling problem. *Information Sciences*, *397-398*, 137 – 154.

Zhao, Z.-J., He, X.-Q., & Liu, F. (2017). An improved multi-objective memetic algorithm for bi-objective permutation flow shop scheduling. In *14th international conference on services systems and services management, icsssm 2017 - proceedings.*