Contents lists available at GrowingScience

Journal of Project Management

homepage: www.GrowingScience.com

Earliness/tardiness minimization in a no-wait flow shop with sequence-dependent setup times

Andrés Felipe Guevara-Guevara[a], Valentina Gómez-Fuentes[a], Leidy Johana Posos-Rodríguez[a], Nicolás Remolina-Gómez[a] and Eliana María González-Neira[b*]

[a]Research assistant, Departamento de Ingeniería Industrial, Facultad de Ingeniería, Pontificia Universidad Javeriana, Carrera 7 No. 40-62, Bogotá, D.C., Colombia
[b]Assitant Professor, Departamento de Ingeniería Industrial, Facultad de Ingeniería, Pontificia Universidad Javeriana, Carrera 7 No. 40-62, Bogotá, D.C., Colombia

C H R O N I C L E

Article history:
Received: October 3, 2021
Received in revised format: October 20, 2021
Accepted: December 26, 2021
Available online:
December 29, 2021

Keywords:
No-wait flow shop
Earliness, tardiness
Genetic algorithm
Just in time
Sequence-dependent setup times

A B S T R A C T

The no-wait flow shop scheduling problem (NWFSP) plays a crucial role in the allocation of resources in multitudinous industries, including the steel, pharmaceutical, chemical, plastic, electronic, and food processing industries. The NWFSP consists of n jobs that must be processed in m machines in series, and no job is allowed to wait between consecutive operations. This project deals with NWFSP with sequence-dependent setup times for minimizing earliness and tardiness. From the literature review of the last five years in NWFSP, it is noticeable that only around 1.92% of the researchers have studied that multi-objective function, which could help to improve the productivity of industries where methods such as just in time are considered. Besides, there is no information about previous researchers that have solved this problem with sequence-dependent setup times. Firstly, a MILP model is proposed to solve small instances, and secondly, a genetic algorithm (GA) is developed as a solution method for medium and large instances. Compared with the mathematical model for small instances, the GA obtained the optimal solution in 100% of the cases. For medium and large instances, the GA improves in an average of 31.54%, 38.09%, 44.58%, 47.72%, and 37.33% the MDD, EDDP, ATC, SPT, and LPT dispatching rules, respectively.

© 2022 Growing Science Ltd. All rights reserved.

1. Problem statement and justification

Flow Shop Scheduling Problem (FSP) is a significant current research area and has been widely studied since it was proposed by Johnson (1954). FSP consists of m machines in series, in which n jobs must be processed (Gupta & Stafford, 2006). The no-wait FSP (NWFSP) is a special case in which n jobs are processed in the same order on m machines, and no job is allowed to wait between consecutive operations until the whole process is done. Thus, the starting time of a job on the first machine might be delayed due to the no-wait conditions. This problem has important applications in multitudinous industries, including the steel, pharmaceutical, chemical, plastic, electronic, and food processing industries (Sapkal & Laha, 2013). The modern agile manufacturing system, in which robots and industrial machines implement a highly coordinated process, can be modeled as an NWFSP (Bertolissi, 2000).

* Corresponding author.
E-mail address: eliana.gonzalez@javeriana.edu.co (E. M. González-Neira)

© 2022 Growing Science Ltd. All rights reserved.
doi: 10.5267/j.jpm.2021.12.001

From the literature review of the last five years in NWFSP, it can be highlighted that most papers have considered one objective function, specifically the makespan (around 53,85% of the studies). Although the makespan improves the machine's utilization, other measures related to time deliveries are essential to improve service level and optimize resource consumption (i.e. reduce energy costs). This importance relies on the evolution of supply chain management, which has made customers and suppliers look for high-quality coordination in their operations (Schaller & Valente, 2020), causing them to adopt Just in Time (JIT) methods. JIT philosophy aims to identify and eliminate waste components such as overproduction, waiting time, transportation, inventory, movement, and defective products. In a traditional JIT production environment, the scheduling problem focuses not only on minimizing tardiness but also on reducing earliness (Joanna Józefowska, 2007; Pinedo, 2008). Jobs that are completed earlier than their due date may cause such opportunity costs, deterioration of the product, and inventory holding costs. Otherwise, tardiness may cause missing customers, contract penalties, loss of sales, and loss of reputation (Rad et al., 2015). Another aspect to be highlighted from the literature is that most studies have analyzed the NWFSP in its standard condition (around 55,77%). Otherwise, foregrounding the papers that take one or more additional constraints, compared to the standard problem, it is noticeable that the allocation of limited resources and idle time are two aspects widely studied in the literature. According to Allahverdi (2015), considering constraints as setup times in manufacturing scheduling problems plays a vital role in delivering reliable products on time. Setup times can be classified as independent and dependent. Independent setup time depends solely on the job to be processed, regardless of its preceding job. Dependent setup times rely on the processing sequence between two consecutive jobs (Allahverdi & Soroush, 2008). However, only 11,76% of the literature has included dependent setup times to the problem despite its importance. Fig. 1shows the distribution of the features studied in the last five years related to NWFSP.

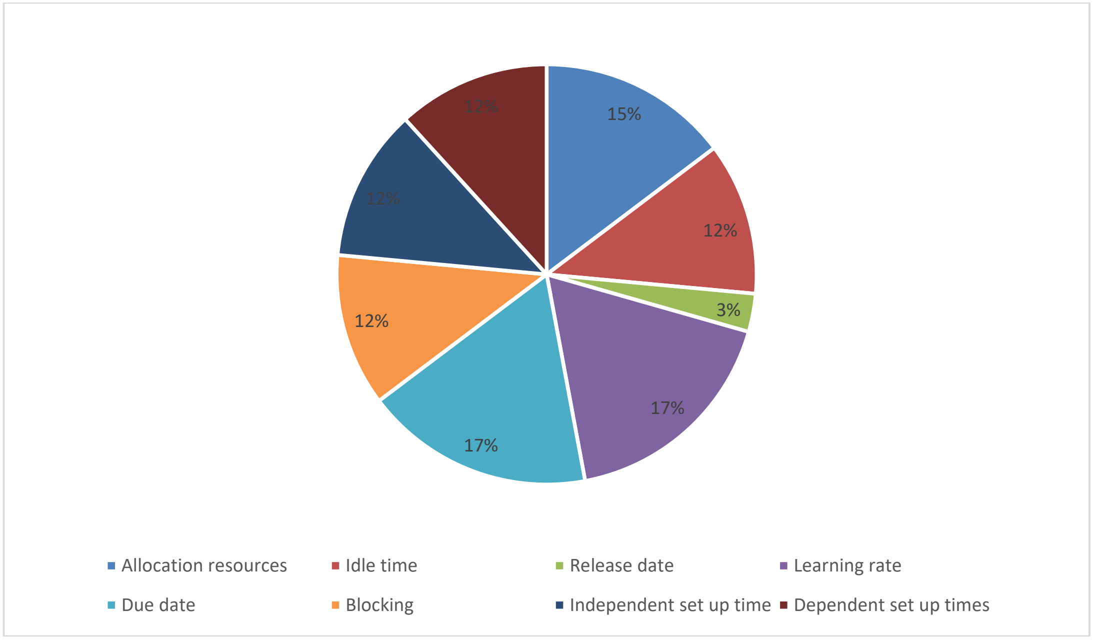

Fig. 1. Features of the NWFSP studied in the last five years. Made by the authors

Concerning the solution methods, a wide variety of mathematical models have been proposed to solve the NWFSP, many of which correspond to meta-heuristics that have shown satisfactory computational results in terms of solution quality, robustness, and efficiency (Cheng, Ying, Li, & Hsieh, 2019). The most remarkable meta-heuristics proposed for the NWFSP have been simulated annealing algorithm (SA) (Aldowaisan & Allahverdi, 2003), tabu search (TS) (Grabowski & Pempera, 2005), particle swarm optimization (PSO) (Liu et al., 2006), variable neighborhood search (VNS) (Komaki & Malakooti, 2017; Schuster & Framinan, 2003) and genetic algorithm (GA) (Aldowaisan & Allahverdi, 2003). GA performs probabilistic search techniques that simulate the process of evolution with the principles of genetics, and it can be applied to solve any problem whose solution space can be represented by a population of structures. It has allowed GA to solve scheduling problems, where the sequences of jobs can be represented as structures in a population (Nouri et al., 2019).

Considering the elements previously mentioned, this project aims to solve the NWFSP with sequence-dependent setup times for minimizing earliness and tardiness through an accurate GA capable of solving small, medium, and large size instances.

The rest of the paper is organized as follows. Section 2 presents the state of the art in NWFSP. In Section 3, a mixed linear integer programming model is explained. Section 4 describes the proposed genetic algorithm for solving medium and large-size instances. Section 5 provides the computational experiments performed and the analysis of results. Conclusions are presented in Section 6.

2. State of the art

The first study of the FSP relating to the no-wait constraint was proposed by (Gilmore & Gomory, 1964), who used an O (n log n) time algorithm to make an application to a two-machine case. Then, a study for three or more machines was conducted by (Reddi & Ramamoorthy, 1972). Since then, many other mathematical models have been developed to solve the NWFSP. A mixed-integer goal programming model to solve a multi-objective NWFSP was proposed by (Selen & Hott, 1986). (Lee & Jung, 2005) provided a mixed-integer linear programming (MILP) model for the NWFSP with sequence-dependent setup times (SDSTs), where there are priority constraints. (Samarghandi, 2015) proposed a MILP model investigating the NWFSP with hard due date constraints, which could solve a few small problems to minimize the makespan. (Ying, Lu, & Lin, 2018) also considered the NWFSP with hard due date constraint and improved upon the approaches developing a new-fangled MILP model and a two-phase enumeration algorithm. More recently, (Schaller & Valente, 2020) developed a branch and bound algorithm to minimize the total earliness and tardiness, only suitable for small-sized instances.

The exact algorithms have shown their effectiveness for solving small-sized problems. Nevertheless, as the problem size increases, the total time for solving the problem becomes longer. This difficulty of solving large instances is because the NWFSP with more than two machines is strongly NP-hard, i.e. the NWFSP becomes more challenging and difficult to solve as the problem size increases (Garey & Johnson, 1979). That is why heuristics and meta-heuristics have been used widely for solving this type of combinatorial problem. Noticeable constructive heuristics have been developed for solving the NWFSP over the last few decades. A slope matching (S/M) method, which is based on geometrical relationships among the cumulative process times, was created by (Bonney & Gundry, 1977). The RAJ heuristic (takes its name from the author's initials), which employs heuristic preference relations and job insertion, was presented by Rajendran (1994). Moreover, many other constructive heuristics for the regular flow shop were also designed to solve the NWFSP. For instance, (Nawaz, Enscore, & Ham, 1983) proposed the NEH heuristic, which has shown a good performance in no large FSP problems. Otherwise, several remarkable meta-heuristics have also been developed for solving the NWFSP. The discrete particle swarm optimization (DPSO) was designed by (Pan, Tasgetiren, & Liang, 2008) and (Pan, Wang, et al., 2008) for solving the NWFSP considering both makespan and total flowtime criteria. Speed-up methods were proposed for the swap and insert neighborhood structure. The improved iterated greedy algorithm (IIGA) was proposed by (Pan, Tasgetiren, et al., 2008; Pan, Wang, et al., 2008) for solving the NWFSP. Two constructive heuristics, which are named improved standard deviation heuristic (ISDH) and improved Bertolissi heuristic (IBH), were presented. Four integrated heuristics were obtained by combining the standard deviation heuristic and the Bertolissi heuristic with the procedure of the constructive heuristic of Laha. A heuristic for NWFSP was introduced by (Sapkal & Laha, 2013). A tabu-mechanism improved iterated greedy (TMIIG) algorithm was proposed by (Ding et al., 2015). In the TMIIG, a tabu-based construction strategy and vast neighborhood structures are applied to improve the quality of solutions.

An effective and efficient heuristic for no-wait flow shop production to minimize total completion time called Current and Future Idle (CFI) that consists of three phases: phase 1 for initial sequence generation, phase 2 for the insertion and neighborhood exchanging, and phase 3 for iteration improvement, was developed by (Ye et al., 2017). An approach to the NWFSP with makespan minimization based on the Ant Colony Algorithm (ACO) was developed by (Riahi & Kazemi, 2016) to generate the initial solution, followed by the SA as a local search method, which differs from the most local search heuristics because it uses two or more neighborhoods, instead of one, in its structure. A Self-adaptive ruin-and-recreate algorithm was presented by (Ying et al., 2016). An initial solution with the famous NEH (Nawaz et al., 1983) heuristic is implemented, which is then iteratively improved by a self-adaptive ruin-and-recreate procedure with a speed-up method. The current self-adaptive mechanism dynamically fixes the size and range of the neighborhood during the ruin phase, thus enabling the incumbent solution to escape from the local minima, while the proposed speed-up method accelerates the evaluation of the neighborhoods.

Qi et al. (2016) proposed a fast-local neighborhood search (FLNS) algorithm to minimize the total flow time. They proposed as an initial solution an unscheduled job based on the total processing time and standard deviation of jobs machines. This job sequence is submitted as an initial optimization using a basic neighborhood search algorithm. Later, an innovative local neighborhood search scheme was developed to search the partial neighborhood in each iteration and calculate the neighborhood solution with an objective increment method. An efficient hybrid Particle Swarm Optimization (PHPSO) metaheuristics algorithm was developed by (Bewoor et al., 2017) with the objective of total flow time minimization. This algorithm initializes population through the NEH heuristic and applies an evolutionary search guided by the mechanism of PSO, in the same way, a simulated annealing metaheuristic based on a local neighborhood search to prevent getting stuck in a local optimum. Furthermore, a solution to NWFSP using the flower pollination algorithm based on the hormone modulation mechanism was proposed by (Qu et al., 2018), where random keys are encoded based on an ascending sequence of components to make the flower pollination algorithm (FPA) suitable for the NWFSP. An improved discrete migrating birds optimization (IDMBO) algorithm was presented by (Zhang et al., 2020) to solve NWFSSP with makespan criterion. The standard deviation heuristic (SDH) is performed to generate the initial solution. The different formations of hybrid multi- neighborhood strategy, which includes four neighborhood structures based on insertion and swap operators, are the key to the effectiveness and efficiency, as they generate a deeper search and avoid a local optimum.

Table 1 presents the studies of single objective and multi-objective NWFSP, which have been studied in the last five years, as a percentage of 52 articles researched by the authors. To have a clear vision of the themes that have not been studied as much as others at the date, and finally, make an approach to the investigation work. As can be seen, most studies considered one objective function (75%), where more than half of them have studied the NWFSP to minimize makespan. Flowtime was analyzed only by approximately one-sixth part of the single objective research. Other objectives such as total tardiness, total absolute deviation of job completion times, flow time, resource consumption, conditional value-at-risk (CVaR) of the residual work content, and weighted number of just-in-time jobs were studied in only one work separately. One-quarter part of the research focused on multiple objectives. It is noticeable that the most studied multi-objective NWFSP analyzing the minimization makespan and total tardiness (7.69%) followed by total tardiness and resource consumption which represents 5.77% of the NWFSP studied in the last five years. Furthermore, only 3.85% of the works were focused on studying the makespan and flow time at the same time. Ultimately, only 1.92% of the previous researchers have considered problems such as minimize makespan and idle time, minimize makespan and total system utilization time, minimize makespan and resource consumption and minimize total tardiness and earliness.

Table 1
Objective and multi-objective NWFSP studied in the last five years. Made by the authors

Single/Multiple Objective	Description	%
Single-Objective	Minimize makespan	53.85%
	Minimize flow time	11.54%
	Minimize total tardiness	1.92%
	Minimize the total absolute deviation of job completion times (SATDC)	1.92%
	Minimize resource consumption	1.92%
	Maximize the weighted number of just in-time jobs	1.92%
	Minimize the conditional value-at-risk (CVaR) of the residual work content	1.92%
Multi-Objective	Minimize makespan and total tardiness	7.69%
	Minimize total tardiness and resource consumption	5.77%
	Minimize makespan and flow time	3.85%
	Minimize total tardiness and earliness	1.92%
	Minimize makespan and idle time	1.92%
	Minimize makespan and utilization time	1.92%
	Minimize makespan and resource consumption	1.92%

3. Mixed integer linear programming model

In this section, a MILP model for the NWFSP with sequence-dependent setup times that minimized earliness and tardiness is developed. The nomenclature used is shown below, namely sets, parameters, and decision variables. The objective function and constraints are characterized subsequently.

Sets
$J = jobs$
$I = machines$
Parameters
n = number of jobs.
m = Number of machines.
$p_{ji} = Processing\ time\ of\ job\ j \in J\ on\ machine\ i \in I$
$S_{jki} = Setup - time\ of\ job\ j \in J\ on\ machine\ \ i \in I\ if\ job\ k \in J\ is\ the\ immediately\ preceding\ job.$
$d_j = Due\ date\ of\ job\ j \in J$
Q = Big positive number

Decision Variables
$E_j = \ earliness\ of\ job\ j \in J$
$T_j = \ Tardiness\ of\ job\ j \in J$
$X_{jk} = \{1\ if\ job\ j \in J\ is\ processed\ after\ job\ k \in J, 0\ otherwise\}$
$C_{ij} = Completion\ time\ of\ job\ j \in J\ on\ machine\ i \in I$
$\mathrm{ST}_{ji} = Starting\ time\ of\ job\ j \in J\ on\ machine\ i \in I$

Objective Function

$$\min Z = \sum_{j\in J} E_j + T_j \tag{1}$$

Constraints

$$\sum_{j\in J} x_{0j} = 1 \tag{2}$$

$$\sum_{k\in J} x_{jk} = 1 \quad \forall j \in J \tag{3}$$

$$\sum_{j\in J} x_{jk} \leq 1 \quad \forall k \in J \tag{4}$$

$$x_{jk} + x_{kj} \leq 1 \quad \forall j,k \in J \tag{5}$$

$$C_{ji} = \mathrm{ST}_{ji} + p_{ji} \qquad \forall i \in I\,, \qquad \forall j \in J \tag{6}$$

$$\mathrm{ST}_{ji} \geq \mathrm{C_{ki}} + \mathrm{S_{jki}} - \left(\left(1 - \mathrm{X_{kj}}\right) * \mathrm{Q}\right) \quad \forall i \in I, \forall j,k \in J, j \neq k \tag{7}$$

$$\mathrm{ST}_{ji} = \mathrm{C_{j,i-1}} \qquad \forall j \in J, \qquad \forall\ i \in I, i \geq 2 \tag{8}$$

$$T_j \geq C_{j|I|} - d_j \qquad \forall j \in J \tag{9}$$

$$E_j \geq d_j - C_{j|I|} \qquad \forall j \in J \tag{10}$$

$$\mathrm{E_j} \geq 0 \quad \forall j \in J \tag{11}$$

$$\mathrm{T_j} \geq 0 \quad \forall j \in J \tag{12}$$

$$\mathrm{C_{ji}} \geq 0 \quad \forall j \in J, \qquad \forall i \in I \tag{13}$$

$$x_{\mathrm{jk}} \in \{0,1\}\ \forall i \in I, \forall j \in J \tag{14}$$

The objective function (1) is the minimization of total earliness and tardiness. Constraint set (2) generates one dummy job (job 0), to initialize the sequence in the first machine. Constraint set (3) ensures that each job must have exactly one predecessor. Constraint set (4) indicates that every job has no more than one succeeding job. Constraint set (5) guarantees that a job might not be both predecessor and successor of another job at the same time. Constraint set (6) asserts that the completion time of a job is its starting time on a machine added to its processing time on the same machine. This constraint set makes certain the no-wait constraint. Constraint set (7) shows that the starting time in the following job is wider than the total sum of the setup time and processing time of the job processed as the latter. Constraint set (8) guarantees that the starting time of a job on a machine is equal to the completion time in the previous machine. Constraint set (9) formulates the tardiness of every job. Constraint set (10) formulates the earliness of every job. Constraint sets (11), (12), (13), and (14) define the domain of the decision variables.

4. Proposed Genetic algorithm (GA)

According to Ruiz and Allahverdi (2007a), a GA is a bio-inspired method that has shown excellent performance in various scheduling problems. The algorithm starts by generating a primitive population of chromosomes (solutions) while satisfying the limits and constraints of the problem. The chromosomes are deduced from successive replicates called generations. During each generation, these chromosomes are evaluated according to the fitness function (optimization objective), and the chromosomes with the best fitness function have more probabilities of reproduction. The basic iterative structure of the proposed algorithm is presented in Pseudocode 1.

Pseudocode 1. Genetic Algorithm for the NWFSP

```
Begin /*Genetic Algorithm*/
Initialize the parameters -> Population size, mutation probability, maximum execution time , and number of generations
Generate initial population
Calculate the objective function and the fitness value for each individual
Do While Generation < Last generation and Execution time < Maximum execution time
        Increase generation counter: g-> g+1
        Select the parents with the criteria of their probability
        Crossover the parents randomly and create the offspring population
        Generate random number to apply the mutation probability for offspring population
        If Random<=Mutation probability, then
                Mutate children
        Evaluate the fitness value of the children
End While
Return best solution
End
```

4.1. Chromosome representation

Usually, in flow shop scheduling problems, job-based representation is a common scheme to encode a solution, where a single chromosome represents a sequence, i.e., the first element is the first job to be processed, the second element is the second job to be processed, and so on. In Fig. 2 an example of the chromosome for a flow shop with six jobs is presented, indicating that the first job to be processed is job 6, and the final job to be processed is job 2.

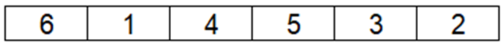

Fig. 2 Chromosome representation. Made by the authors.

4.2. Initial population

Traditional GA starts with randomly generated chromosomes. However, to improve the approach and obtain better solutions, some of the most remarkable dispatching rules, both static and dynamic, presented in the literature for different shop environments, were adapted to the problem and added to the initial population. Most precisely, the dispatching rules considered rules were:

- ➢ Modified due date (MDD): Jobs are scheduled once at a time, every time the system becomes available (t). At the time t, the job with the minimum value of $\max\{d_j, C_{j|I|}\}$ is selected.
- ➢ Earliest due date with processing (EDDP): Jobs are sorted in ascending order according to this index $\frac{d_j}{\sum_i p_{ij}}$
- ➢ Shortest processing time (SPT): Jobs are sorted in ascending order according to their processing times. To adapt the rule to the problem, the processing time of each job is assumed as the sum of its processing times in all the machines.
- ➢ Longest processing time (LPT): Elements are assigned from the job with the longest to the shortest processing time.
- ➢ Apparent tardiness cost (ATC): Jobs are scheduled once at a time, every time the system becomes available (t). At the time t, the job with the highest value of $\frac{w_j}{p_j}\exp\left[-\frac{\max(d_j - p_j - t, 0)}{k\bar{p}}\right]$ is selected. Here w_j is the weight assigned for each job, which in this case is the same for all of them, $\bar{p}$ is the average processing time of all remaining jobs and k is a look-ahead parameter calculated as indicated in equations (15-17):

$$k = 4.5 + R, for\ R \leq 0.5 \tag{15}$$

$$k = 6 + 2R, for\ R > 0.5 \tag{16}$$

where:

$$R = \frac{d_{max} - d_{min}}{C_{max}} \tag{17}$$

In this case as in the previous rules, p_j refers to the sum of the processing times of the job in all the machines.

4.3. Parents selection

The crucial point in the performance of a GA is the selection of individuals based on their fitness value to generate a population for the next generation (Du & Swamy, 2016). Following the roulette selection, a probability proportional to the fitness value of the chromosome is assigned, which determines the individuals that have a greater chance of reproducing.

4.4. Crossover

Once the parents are selected, the chromosomes are crossed over. For this purpose, the sequence of the first parent is replicated until a position is determined by a random number C previously generated between 1 and the total number of jobs. The remaining positions are filled with those jobs that have not been considered yet by arranging them in the same order as they are positioned in the sequence of the second parent. Then, a second child is also generated using the same method but reversing the order of the parents. This crossover process is illustrated in Fig. 3.

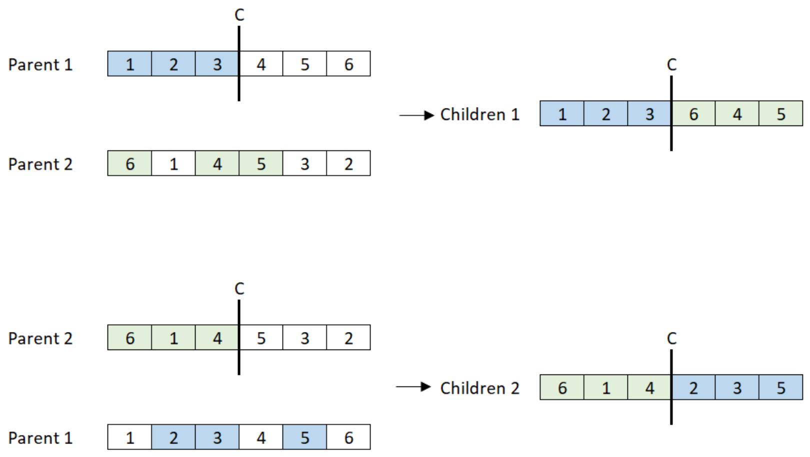

Fig. 3. Crossover representation. Made by the authors.

4.5. Mutation

For each generation, the offspring may or may not mutate. This decision is determined by a mutation probability parameter. For each chromosome, a random number is generated. If it is lower than the mutation probability, then that child is selected for mutation. For this mutation, two jobs r1 and r2, are randomly chosen and their positions are switched as in Fig. 4.

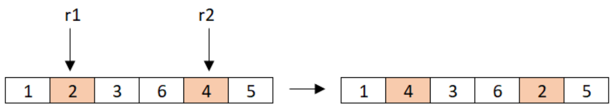

Fig. 4. Mutation representation. Made by the authors.

4.6. Objective function evaluation

The objective function was calculated in two phases. In the first phase, the completion times of jobs are computed, and then the earliness plus tardiness function is obtained. In the second phase, an adjustment is made to the starting times of the jobs by moving them as much as is possible without affecting the sequence but improving the objective function value. This adjustment is made in the sequence from the end to the beginning, i.e., schedule development starts by placing the last job to be processed and it continues positioning jobs adjacently until it reaches the first job to be processed. Note that the first fixed job will always finish its processing in time instant equivalent to its due date. Once the sequence is adjusted, the tardiness and earliness are calculated again, obtaining in this way the objective function.

5. Computational experiments

This section presents all experiments performed to evaluate the proposed GA, and it is divided into four parts: i) creation of small, and medium and large instances; ii) parametrization of GA, iii) comparison of GA versus MILP model for small instances, and iv) comparison of GA versus some well-known dispatching rules for medium and large instances. All experiments were executed on a personal computer with an AMD Ryzen 5 3500U 2.1GHz processor and 8GB RAM.

5.1. Instances

5.1.1. Small instances

A total of 12 small instances were created. Those were divided into two groups: 1) tight due dates with a narrow range, with tardiness factor (T) of 0.5 and due date range (R) of 0.5; and 2) loose due dates with a wide range, with T=0.3 and R=0.9. In both groups, instances of the same size were tested, which can be shown in Table 2.

Table 2
Groups of small instances. Made by the authors.

Group 1: T=0.5 and R=0.5 Group 2. T=0.3 and R=0.9
3 machines and 4 jobs
4 machines and 3 jobs
4 machines and 6 jobs
6 machines and 12 jobs
6 machines and 15 jobs
6 machines and 18 jobs

For each instance, the processing times, setup times, and due dates were randomly generated from specified probability distributions. First, the processing times were generated from a normal distribution with a mean of 150 and a standard deviation of 2. Likewise, the setup times were generated through a normal distribution with a mean of 100 and a standard deviation of 2. Finally, a critical issue was the due date of the jobs. In the scheduling literature, it is common that due dates be generated randomly according to a uniform distribution. The expression that is commonly used is: [P(1−T−R/2), P(1−T+R/2)] where P is a lower bound of the makespan of a given instance, and T and R are respectively the tardiness factor and due date range. We considered two combinations of T and R, 0.3–0.9 (looser and more spaced due dates) and 0.5–0.5 (tight and clustered due dates). The tardiness factor is an indicator of the tightness of due dates, and the due date range is a measure of the dimension of the due date ranges of an instance. We proposed the calculation of P as an adaptation of the lower bound proposed by Ruiz and Allahverdi (2007b). The adjustment consists, for the first part of the equation, in using the shortest time between all pair of jobs in the first machine; later, in the last part of the equation, we selected the maximum time of the whole three-dimensional matrix for the dependent setup time, the Eq. (18) is shown as follow:

$$P = LB(Cmax) = \min_{j=1,\dots,n}\left(\min_{k=1,\dots,n}(s_{jk0}) + \sum_{i=1}^{m-1} p_{ji}\right) + \left(\sum_{j=1}^{n}\sum_{i=1}^{m} p_{ji} + int\left(\frac{\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{i=1}^{m} s_{jki}}{|J| * |K| * |I|}\right)\right) - \max(s_{jki}) \quad (18)$$

5.1.2. Medium and large instances

We used 3000 benchmark instances taken from the paper “No-wait flowshop with separate setup times to minimize maximum lateness” written by Ruiz and Allahverdi (2007b). To these instances, we modified the setup times to sequent-dependent setup times. The benchmark instances resulted from the combinations of factors presented in Table 3, and there were 10 instances per combination.

Table 3
Factors for creation of medium and large size instances. Made by the authors

Factor	Symbol	Number of levels	Values
Number of jobs	j	5	20, 40, 60, 80, 100
Number of machines	i	5	3, 5, 10, 15, 20
Combination of tardiness factor (T) and due date range (R)	TR	2	T0.3 R0.9, T0.5 R0.5
Distribution of the setup times as a percentage of the processing times	DS	3	U[0,50]%, U[0,100]%, U[0,150]%,
Size of Processing times	SP	2	U[1,5], U[1,10]

5.2. Parametrization of GA

An experimental design was carried out to determine which values of mutation probability, population size, and stopping time factor presented the best performance of the GA. Four instances from those mentioned in subsection 5.1.2 were selected at random to set the parameters of GA. The factors analyzed in the experiment were instances = {1,2,3,4}, probability of mutation (Pmut) = {0.01, 0.02, 0.03}, population size (Pop) = {10, 30, 50} and stopping time factor (TimeFactor) = {0.5s, 1.1s, 1.7s}. This implies 108 treatments. For each treatment, 2 observations were taken. The response variable was the improvement percentage of the objective function obtained by GA in comparison with the objective function obtained by applying the MDD dispatching rule. It is important to note that the stopping factor is a factor that is multiplied by the number of jobs and number of machines of the instance to obtain the stopping time criterion of the GA, i.e. ($NumberJobs \cdot NumerMachines \cdot TimeFactor$) in seconds. According to the ANOVA, with a significance of 0.05, there is a significant effect of mutation probability, instance size, and stopping time in the algorithm's performance. Additionally, there is a significant effect of some interactions. It is important to remark that in the ANOVA three assumptions must be accomplished: normality, homoscedasticity, and independence of the residuals. Normality was evaluated with Shapiro Wilks, homoscedasticity with Levene test, and independence with Durbin Watson test. Only the independence assumption was fulfilled with a p-value of 0.83 whereas normality and homoscedasticity were not fulfilled with p-values < 0.05. Therefore, a non-parametric test for factorial designs called ANOVA-type statistics, proposed by Brunner et al. (Brunner et al., 1997), was performed. The results of the ANOVA-type statistic can be seen in Table 4. This test also provides confidence intervals for the rankings allowing to select those combinations that present the best performance.

Table 4
ANOVA-Type Statistic for parametrization of GA. Made by the authors

	Statistic	df1	df2	p-Value
Pmut	8.158965531	1.992678009	144.7059908	0.000446611
Pop	0.779029429	1.992772099	144.7059908	0.460345902
Pmut:Pop	0.19982244	3.969267397	144.7059908	0.937141073
TimeFactor	129.4712879	1.81764231	144.7059908	0
Pmut:TimeFactor	0.559185966	3.580868498	144.7059908	0.673474174
Pop:TimeFactor	0.28187083	3.606974876	144.7059908	0.872025075
Pmut:Pop:TimeFactor	0.201649992	7.090744102	144.7059908	0.985285259

After analyzing the confidence intervals of rans provided by the ANOVA-Type test the best treatment was Pmut = 0.03, Pop = 50 and TimeFactor=1.1, which is the configuration used to run all the instances generated in sections 5.1.

5.3. Evaluation of the performance of GA

In this subsection, two computational experiments were conducted to assess the efficiency of the proposed GA. Firstly, the comparison of the GA results with the MILP model for small instances, and secondly, the comparison of GA with some well-known dispatching rules for medium and large instances.

5.3.1. Performance of GA vs. MILP model in small instances

The proposed GA and MILP model were tested for the small instances presented in subsection 5.1.1. The MILP model was executed with a maximum time of 10000s.

Table 5
Computational results of small instances

		Combination of tardiness factor (T) and due date range (R) - TR		CPU Times (s)		Objective Function (Earliness + Tardiness)	
Machines	Jobs	T	R	MILP	GA	MILP	GA
3	4	0.3	0.9	0.363	21.2	178*	178*
3	4	0.5	0.5	0.346	21.2	150*	150*
4	3	0.3	0.9	0.245	21.2	1539*	1539*
4	3	0.5	0.5	0.240	21.2	640*	640*
4	6	0.3	0.9	0.368	34.4	216*	216*
4	6	0.5	0.5	0.369	34.4	982*	982*
6	12	0.3	0.9	2158.8	87.2	722*	722*
6	12	0.5	0.5	137.9	87.2	67*	67*
6	15	0.3	0.9	10000	107	1995	869
6	15	0.5	0.5	1994	107	149*	149*
6	18	0.3	0.9	10000	126.8	3993	1331
6	18	0.5	0.5	10000	126.8	12860	797

*Optimal solutions

Table 5 shows the results for each combination of machines, jobs, due date range, and tardiness factor. Results indicate that the time required to solve instances with the mathematical model increase rapidly when the number of machines and jobs increases, thus it is noticeable that one instance of 6 machines and 15 jobs and both instances of 6 machines and 18 jobs cannot even get an optimal solution before 10000s. There seems to be an increase in CPU times when the number of jobs augments. We also looked at how the due date affected the results. It had a larger effect in the objective function than in CPU times on both the MILP and the proposed genetic algorithm. Most of the time as the due date range is wider, the value of the objective function increases along with it. In addition, our heuristic reaches 9 of 9 optimums and it significantly improves the objective function, in much less time, of those three instances in which the MILP could not obtain optimal solution in 10000s.

5.3.2. Performance of GA vs. dispatching rules in medium and large instances

To the best of our knowledge, this is the first time the NWFSP with sequence-dependent setup times to minimize earliness and tardiness is considered. Therefore, there are no benchmark results from the literature that we can compare against. Then, we compare the results of the proposed GA with MDD, EDDP, SPT, LPT, and ATC well-known dispatching rules. The measure was the percentage of improvement of the objective function, see Eq. (19). The results, presented in Table 6, demonstrate that the GA has a better performance compared to every single dispatching rule considered.

$$\% \text{ improvement} = \frac{FO.dispatching\ rule - \mathrm{FO.GA}}{FO.dispatching\ rule} \tag{19}$$

Table 6
Improvement average resume of the GA

Machines	**Jobs**	**Dispatching rule**				
		MDD	EDDP	ATC	SPT	LPT
3	20	36.68%	50.09%	47.76%	61.75%	64.70%
	40	33.99%	41.25%	47.80%	53.13%	56.41%
	60	34.40%	35.90%	41.16%	42.56%	46.14%
	80	33.49%	35.27%	32.87%	33.37%	40.66%
	100	32.33%	33.47%	27.20%	27.64%	34.55%
5	20	36.39%	50.43%	44.36%	64.65%	65.41%
	40	36.53%	42.81%	47.52%	54.07%	57.61%
	60	34.17%	36.38%	43.58%	45.17%	47.37%
	80	33.90%	35.02%	35.21%	35.84%	41.06%
	100	34.35%	35.61%	33.02%	33.04%	37.12%
10	20	31.26%	47.52%	39.95%	62.10%	63.90%
	40	31.40%	40.26%	44.87%	53.51%	56.88%
	60	31.70%	33.36%	40.63%	42.16%	44.57%
	80	32.44%	33.00%	37.07%	37.65%	40.10%
	100	33.74%	33.43%	33.22%	33.42%	36.77%
15	20	29.02%	47.08%	36.39%	61.53%	63.43%
	40	29.50%	38.04%	44.61%	52.94%	55.25%
	60	30.24%	31.93%	39.70%	41.90%	44.54%
	80	31.89%	31.82%	35.64%	36.37%	39.82%
	100	31.73%	32.64%	33.44%	33.86%	36.16%
20	20	30.89%	48.26%	38.63%	62.78%	64.81%
	40	31.80%	38.50%	46.06%	54.71%	56.49%
	60	29.48%	31.36%	40.58%	42.89%	45.68%
	80	29.89%	30.40%	34.50%	35.34%	38.53%
	100	30.38%	30.91%	31.71%	32.21%	34.25%
Average improvement per dis-patching rule		31.54%	38.09%	44.58%	47.72%	37.33%
Global Average				39.85%		

To analyze if there was any sort of relation between the improvement and the instance characteristics, an experimental design was carried out. The factors were TR of due dates, SP, DS, number of jobs, and number of machines. Due to the assumptions of normality and homoscedasticity were not fulfilled, with p-values < 0.001, the non-parametric test ANOVA-Type Statistic was performed. Table 7 summarizes the results obtained from the ANOVA-Type by each main factor.

Table 7
ANOVA-Type Statistic Results by individual factor of improvement of GA vs Dispatching Rules

Factor	**P value**	**Best performance**
TR	< 0.0001	0.3-0.9
SP	0.000108033	5
DS	< 0.0001	50%
Jobs	< 0.0001	40
Machines	0.64	Does not apply

It can be noticed that the best performance of the proposed GA regarding TR corresponds to the instances with looser and more spaced due dates. Besides, when the size of processing times are generated with U[1,5] the GA improves in more quantity the results of dispatching rules. Also, the GA showed a better improvement vs dispatching rules with the lowest setup times DS, which means, the setups with a maximum value of 50% of the processing times. Additionally, instances with 40 jobs demonstrated to have the best results for the execution of the GA concerning the number of jobs. The only factor from which the improvement value resulted to be independent, was the number of machines, in other words, the GA has the same level of performance no matter how big the problem is in terms of the number of machines. Nevertheless, there are some significant interaction effects in which machines are involved. The results of the interactions between two factors where the GA had better performance are illustrated in Table 8.

Table 8
The best performance combinations

Double interaction		P-value	Combinations with best performance of the GA	
TR	SP	0.7750	0.3-0.9	5
TR	DS	0.6156	0.3-0.9	50%
TR	Jobs	0.0849	0.3-0.9	40
TR	Machines	0.0000	0.3-0.9	15
			0.3-0.9	20
			0.3-0.9	10
SP	DS	0.7280	5	50%
			10	50%
SP	Jobs	0.0001	5	40
			10	40
			10	20
SP	Machines	0.0880	SP 5	
DS	Jobs	0.0012	50%	40
			100%	40
			50%	20
DS	Machines	0.0000	50%	20
			50%	15
			150%	3
Jobs	Machines	0.0000	40	5
			40	3
			20	3

6. Conclusions

This paper has solved the No-wait Flow Shop problem (NWFSP) with sequence-dependent setup times to minimize total earliness and tardiness. We proposed a mixed-integer linear programming model (MILP) to solve small instances and a genetic algorithm (GA) to solve medium and large instances. First, small instances were tested in the mathematical model. The results of the MILP model showed that the larger the instance, the computational time required to solve the problem increase exponentially. Thus, when the instances become larger, it is impossible to obtain a feasible integer solution in less than a considerable time (10.000s).

The proposed GA was tested on medium and large instances, and its performance was evaluated against five well-known dispatching rules MDD, EDDP, ATC, SPT, LPT. After analyzing the results obtained for 3000 different instances sizes, the GA demonstrates a high-quality performance in a reasonable time against the most famous dispatching rules found in the literature with an average improvement of 39.85%.

For future research, it is challenging to solve the NWFSP using another metaheuristic such as particle swarm optimization (PSO), or VNS, to compare their performance against the proposed GA. On the other hand, it would be interesting to consider the variability involved in the problem and treat the NWFSP from a stochastic point of view, making the problem closer to reality.

References

Aldowaisan, T., & Allahverdi, A. (2003). New heuristics for no-wait flowshops to minimize makespan. *Computers and Operations Research*, *30*(8), 1219–1231. https://doi.org/10.1016/S0305-0548(02)00068-0

Allahverdi, A. (2015). The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research*, *246*(2), 345–378. https://doi.org/10.1016/j.ejor.2015.04.004

Allahverdi, A., & Soroush, H. M. (2008). The significance of reducing setup times/setup costs. *European Journal of Operational Research*, *187*(3), 978–984. https://doi.org/10.1016/j.ejor.2006.09.010

Bertolissi, E. (2000). Heuristic algorithm for scheduling in the no-wait flow-shop. *Journal of Materials Processing Technology*, *107*(1–3), 459–465. https://doi.org/10.1016/S0924-0136(00)00720-2

Bewoor, L. A., Chandra Prakash, V., & Sapkal, S. U. (2017). Evolutionary hybrid particle swarm optimization algorithm for solving NP-hard no-wait flow shop scheduling problems. *Algorithms*, *10*(4), 1–17. https://doi.org/10.3390/a10040121

Bonney, M. C., & Gundry, S. W. (1977). Solutions to the constrained flowshop sequencing problem. *Operational Research Quarterly*, *28*(3), 663–670.

Brunner, E., Dette, H., & Munk, A. (1997). Box-Type Approximations in Nonparametric Factorial Designs. *Journal of the American Statistical Association*, *92*(440), 1494–1502. https://doi.org/10.1080/01621459.1997.10473671

Cheng, C. Y., Ying, K. C., Li, S. F., & Hsieh, Y. C. (2019). Minimizing makespan in mixed no-wait flowshops with sequence-dependent setup times. *Computers and Industrial Engineering*, *130*(December 2018), 338–347. https://doi.org/10.1016/j.cie.2019.02.041

Ding, J.-Y., Song, S., Gupta, J. N. D., Zhang, R., Chiong, R., & Wu, C. (2015). An improved iterated greedy algorithm with a Tabu-based reconstruction strategy for the no-wait flowshop scheduling problem. *Applied Soft Computing*, *30*, 604–613. https://doi.org/10.1016/j.asoc.2015.02.006

Du, K.-L., & Swamy, M. N. S. (2016). Search and Optimization by Metaheuristics. In *Search and Optimization by Metaheuristics*. https://doi.org/10.1007/978-3-319-41192-7

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP- completeness*. New York: Freeman.

Gilmore, P. C., & Gomory, R. E. (1964). Sequencing a one state - variable machine: A solvable case of the traveling salesman problem. *IBM Watson Research Center, Yorktown Heights, New York*, *6*(3), 30–32. https://doi.org/10.1063/1.3061193

Grabowski, J., & Pempera, J. (2005). Some local search algorithms for no-wait flow-shop problem with makespan criterion. *Computers & Operations Research*, *32*(8), 2197–2212. https://doi.org/10.1016/j.cor.2004.02.009

Gupta, J. N. D., & Stafford, E. F. (2006). Flowshop scheduling research after five decades. *European Journal of Operational Research*, *169*(3), 699–711. https://doi.org/10.1016/j.ejor.2005.02.001

Joanna Józefowska. (2007). Just-In-Time Scheduling: Models and Algorithms for Computer and Manufacturing Systems. *Just-In-Time Scheduling: Models and Algorithms for Computer and Manufacturing Systems*. https://doi.org/10.1007/978-0-387-71718-0

Johnson, S. M. (1954). Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*. https://doi.org/10.1002/nav.3800010110

Komaki, M., & Malakooti, B. (2017). General variable neighborhood search algorithm to minimize makespan of the distributed no-wait flow shop scheduling problem. *Production Engineering*, *11*(3), 315–329. https://doi.org/10.1007/s11740-017-0716-9

Lee, Y. H., & Jung, J. W. (2005). New heuristics for no-wait flowshop scheduling with precedence constraints and sequence dependent setup time. *Lecture Notes in Computer Science*, *3483*(IV), 467–476.

Liu, B., Wang, L., & Jin, Y.-H. (2006). An effective hybrid particle swarm optimization for no-wait flow shop scheduling. *The International Journal of Advanced Manufacturing Technology*, *31*(9–10), 1001–1011. https://doi.org/10.1007/s00170-005-0277-5

Nawaz, M., Enscore, E. E., & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, *11*(1), 91–95. https://doi.org/10.1016/0305-0483(83)90088-9

Nouri, F., Samadzad, S., & Ghahremani Nahr, J. (2019). Meta-heuristics algorithm for two-machine no-wait flow-shop scheduling problem with the effects of learning. *Uncertain Supply Chain Management*, *7*(4), 599–618. https://doi.org/10.5267/j.uscm.2019.5.002

Pan, Q. K., Tasgetiren, M., & Liang, Y. C. (2008). A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. *Computers and Operations Research*, *35*(9), 2807–2839. https://doi.org/10.1016/j.cor.2006.12.030

Pan, Q. K., Wang, L., & Zhao, B. H. (2008). An improved iterated greedy algorithm for the no-wait flow shop scheduling problem with makespan criterion. *International Journal of Advanced Manufacturing Technology*, *38*(7–8), 778–786. https://doi.org/10.1007/s00170-007-1120-y

Pinedo, M. L. (2008). Scheduling: Theory, algorithms, and systems. In *Scheduling: Theory, Algorithms, and Systems*. New York. https://doi.org/10.1007/978-0-387-78935-4

Qi, X., Wang, H., Zhu, H., Zhang, J., Chen, F., & Yang, J. (2016). Fast local neighborhood search algorithm for the no-wait flow shop scheduling with total flow time minimization. *International Journal of Production Research*, *54*(16), 4957–4972. https://doi.org/10.1080/00207543.2016.1150615

Qu, C., Fu, Y., Yi, Z., & Tan, J. (2018). Solutions to no-wait flow shop scheduling problem using the flower pollination algorithm based on the hormone modulation mechanism. *Complexity*, *2018*. https://doi.org/10.1155/2018/1973604

Rad, S. T., Gholami, S., Shafaei, R., & Seidgar, H. (2015). Bi-objective Optimization for Just in Time Scheduling : Application to the Two-Stage Assembly Flow Shop Problem. *Quality Engineering and Production Optimization*, *1*(1), 21–32.

Rajendran, C. (1994). A no-wait flowshop scheduling heuristic to minimize makespan. *Journal of the Operational Research Society*, *45*(4), 472–478. https://doi.org/10.1057/jors.1994.65

Reddi, S. S., & Ramamoorthy, C. v. (1972). On the Flow-Shop Sequencing Problem With No Wait in Process. *Operational Research Quarterly*, *23*(3), 323–331. https://doi.org/10.1057/jors.1972.52

Riahi, V., & Kazemi, M. (2016). A new hybrid ant colony algorithm for scheduling of no-wait flowshop. *Operational Research*, *18*(1), 55–74. https://doi.org/10.1007/s12351-016-0253-x

Ruiz, R., & Allahverdi, A. (2007a). No-wait flowshop with separate setup times to minimize maximum lateness. *International Journal of Advanced Manufacturing Technology*, *35*(5–6), 551–565. https://doi.org/10.1007/s00170-006-0726-9

Ruiz, R., & Allahverdi, A. (2007b). No-wait flowshop with separate setup times to minimize maximum lateness. *International Journal of Advanced Manufacturing Technology*, *35*(5–6), 551–565. https://doi.org/10.1007/s00170-006-0726-9

Samarghandi, H. (2015). A particle swarm optimisation for the no-wait flow shop problem with due date constraints. *International Journal of Production Research*, *53*(9), 2853–2870. https://doi.org/10.1080/00207543.2015.1007245

Sapkal, S. U., & Laha, D. (2013). A heuristic for no-wait flow shop scheduling. *International Journal of Advanced Manufacturing Technology*, *68*(5–8), 1327–1338. https://doi.org/10.1007/s00170-013-4924-y

Schaller, J., & Valente, J. M. S. (2020). Minimizing total earliness and tardiness in a nowait flow shop. *International Journal of Production Economics*, *224*(December 2018), 107542. https://doi.org/10.1016/j.ijpe.2019.107542

Schuster, C. J., & Framinan, J. M. (2003). Approximative procedures for no-wait job shop scheduling. *Operations Research Letters*, *31*(4), 308–318. https://doi.org/10.1016/S0167-6377(03)00005-1

Selen, W. J., & Hott, D. D. (1986). A new formulation and solution of the flowshop scheduling problem with no in-process waiting. *Applied Mathematical Modelling*, *10*(4), 246–248. https://doi.org/10.1016/0307-904X(86)90053-3

Ye, H., Li, W., Abedini, A., & Nault, B. (2017). An effective and efficient heuristic for no-wait flow shop production to minimize total completion time. *Computers and Industrial Engineering*, *108*, 57–69. https://doi.org/10.1016/j.cie.2017.04.002

Ying, K. C., Lin, S. W., & Wu, W. J. (2016). Self-adaptive ruin-and-recreate algorithm for minimizing total flow time in no-wait flowshops. *Computers and Industrial Engineering*, *101*, 167–176. https://doi.org/10.1016/j.cie.2016.08.014

Ying, K. C., Lu, C. C., & Lin, S. W. (2018). Improved Exact Methods for Solving No-Wait Flowshop Scheduling Problems with Due Date Constraints. *IEEE Access*, *6*, 30702–30713. https://doi.org/10.1109/ACCESS.2018.2834954

Zhang, S., Gu, X., & Zhou, F. (2020). An Improved Discrete Migrating Birds Optimization Algorithm for the No-Wait Flow Shop Scheduling Problem. *IEEE Access*, *8*, 99380–99392. https://doi.org/10.1109/ACCESS.2020.2997379

© 2022 by the authors; licensee Growing Science, Canada. This is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC-BY) license (http://creativecommons.org/licenses/by/4.0/).