Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

Performance investigation of metaheuristics for the just-in-time single-machine under different time windows and setup restrictions

Miguel Gonçalves de Freitas[a], Alex Paranahyba Abreu[b], Fábio José Ceron Branco[c], Helio Yochihiro Fuchigami[b*] and Rian Tavares de Mello[b]

[a]*Federal University of Goiás, Aparecida de Goiania, GO, Brazil*
[b]*Federal University of São Carlos, São Carlos, SP, Brazil*
[c]*Federal University of Technology - Paraná, Ponta Grossa, PR, Brazil*

C H R O N I C L E

Article history:
Received November 23 2024
Received in Revised Format January 6 2025
Accepted March 14 2025
Available online March 14 2025

Keywords:
Scheduling
Fireworks algorithm
Earliness-tardiness
Time windows
Sequence-dependent setup
Metaheuristics

A B S T R A C T

In this paper, we assess the performance of five metaheuristics for the single-machine under different time windows and sequence-dependent setup times, optimizing the total weighted earliness and tardiness: Iterated Greedy Algorithm (IGA), Artificial Bee Colony (ABC), Bat Algorithm (BA), Particle Swarm Optimization (PSO), and Fireworks Algorithm (FWA). Many real-world situations require delivery in a specific time interval, analogous to optimization problems with a time window in the Just-in-Time philosophy. Also, several practical situations require different time intervals to prepare the environment to process the activities depending on what was immediately done and what will be executed next, characterizing the sequence-dependent setup problem. These cases are common among operations handling materials of diverse colors, different temperatures, or high demands on sterilization requirements. Statistical results highlight the superiority of the FWA, with the best results in all the problem dimensions analyzed, especially in the larger-size instances, with only 1.23% average relative deviation against 61.18% of the known Iterated Greedy algorithm.

1. Introduction

Scheduling problems with due windows are part of the Just-in-Time (JIT) business philosophy. The Just-In-Time (JIT) approach focuses on ensuring that necessary materials are available precisely when needed, while keeping inventory levels to a minimum. This system eliminates inefficient components such as overproduction, waiting times, unnecessary processing, defective products, queues, unnecessary movement, and idle time (Janiak et al., 2015). In practice, JIT measures are vital for businesses, as both early and late production can lead to increased costs – early production raises inventory levels, while late production can result in penalties, order cancellations, or customer loss. Shabtay & Steiner (2012) highlight JIT applications in fields such as the chemical and high-tech industries, aerospace scheduling, and the production of perishable items with deterministic demand, as well as environments without storage capacity. Extensive literature reviews and different JIT scheduling problems are discussed by Alidaee et al. (2021), Fuchigami & Rangel (2018), Fuchigami et al. (2018), Ríos-Solís & Ríos-Mercado (2012), Józefowska (2007), and Baker & Scudder (1990). In both manufacturing and services, jobs are often expected to be completed within a time interval known as a "due window," rather than at a specific due date. The due window generalizes the classic due date concept by defining a time range during which the task must be completed. If a task finishes outside this window – either early or late – penalties for earliness or tardiness are incurred. An example can be seen in an IT company providing video-on-demand services, where a user's device initiates communication with a server to request the transmission of data in real time. In such cases, the time window ensures that transmission delays remain imperceptible (Janiak et al., 2015). This time window concept is particularly important in technology and communications industries, where there is often a small buffer for delays in transmissions.

* Corresponding author
E-mail helio@dep.ufscar.br (H Y Fuchigami)

ISSN 1923-2934 (Online) - ISSN 1923-2926 (Print)
2025 Growing Science Ltd.
doi: 10.5267/j.ijiec.2025.3.004

Although research on due windows emerged decades ago, studies addressing setup times have only recently been explored. Ribeiro et al. (2010) were among the first to examine a single machine problem with job-dependent due windows and sequence-dependent setup times, proving that the problem $1|<e_j,d_j>,s_{ij}|ET_w$ is strongly NP-hard. Even without setup times, scheduling problems involving distinct due windows were already shown to be NP-hard by Koulamas (1996). Khadivi et al. (2025) published a comprehensive literature review considering deep reinforcement learning, emphasizing the challenges in the scheduling problems area. Sequence-dependent setup times refer to the preparation required between two specific tasks, with durations that vary depending on the sequence of jobs. These setups are common in manufacturing and service industries (Fuchigami et al., 2015; Freitas et al., 2022). In manufacturing, sequence-dependent setups may involve tasks such as cleaning or sterilizing between operations that handle materials of different colors, temperatures, or purity requirements. In service industries, analogous setups may involve transportation distances, training, or loading and unloading tasks. Comprehensive reviews on setup times and costs were presented by Allahverdi (2015), Allahverdi & Soroush (2008), and Allahverdi et al. (1999). Considering the different types of due windows, Janiak et al. (2015) provided a detailed investigation of common due windows, job-dependent due windows (also known as distinct due windows), and problems where the due window size is fixed, but its starting time needs to be assigned. In addition to the adaptive genetic algorithm proposed by Ribeiro et al. (2010), other research has also tackled single-machine scheduling with different due windows and sequence-dependent setups. For instance, Zhao & Tang (2010) investigated setup times dependent on past sequences, with deteriorating jobs, aiming to minimize objectives such as makespan, total flow time, and earliness/tardiness.

Penna et al. (2012) introduced a heuristic based on the Greedy Randomized Adaptive Search Procedure (GRASP), Variable Neighborhood Descent (VND), Tabu Search, and Path Relinking to minimize earliness and tardiness penalties. Jula & Kones (2013) examined scheduling problems with time window constraints, optimizing job scores and makespan. Ahmadizar & Farhadi (2015) developed a mixed-integer linear programming (MILP) model and an imperialist competitive algorithm to minimize earliness, tardiness, holding, and delivery costs, considering job release dates and batch deliveries. Ye et al. (2024) proposed a Variable Neighborhood Search (VNS) algorithm for the traveling salesman problem with time windows with three objectives: minimization of cost, completion time, and tour duration. Distinct due windows for single-machine scheduling with sequence-dependent setup times were explored by Rosa et al. (2017), aiming to optimize weighted earliness and tardiness. They proposed an implicit enumeration method and a VNS algorithm. Later, Rosa et al. (2018) compared four versions of the VNS for this same problem. Despite its high relevance, the scheduling problem of optimizing earliness-tardiness with different time windows and sequence-dependent setup times has attracted little attention in the literature. To our knowledge, no studies have applied the Fireworks Algorithm (FWA) to this problem. Given the success of FWA in solving other problems, one motivation for this research is to explore its effectiveness in the context of different due windows and sequence-dependent setups. The Fireworks Algorithm is a metaheuristic, non-nature, and non-bioinspired approach developed by Tan & Zhu (2010), which simulates the explosion of fireworks to conduct a global search. FWA is distinguished by its explosive search mechanism and multiple interacting populations. According to Liu & Qin (2021) and a comprehensive review by Li & Tan (2019), FWA has garnered substantial attention since its introduction and has proven to be highly competitive or superior in various real-world optimization problems.

Numerous FWA applications, variants, and improvements have been documented, including in a book by its creator, Tan (2015). Enhancements for global optimization were proposed by Li et al. (2017), while He et al. (2019) introduced a bi-objective FWA for flow shop problems with sequence-dependent setup times. Pang et al. (2020) implemented an improved version for hybrid flow shops, and Liu & Qin (2021) developed a neighborhood improvement approach for traffic flow prediction. Xu et al. (2020) proposed a multitask FWA to simultaneously optimize multiple jobs. More recently, FWA has been applied to a wide range of problems such as Meng and Tan (2024), Hao et al. (2024), Ding et al. (2025), among others. The basic FWA framework begins with a population of fireworks, representing solutions to the problem. Each firework explodes, creating clouds of sparks that represent new solutions. The explosion's intensity and radius depend on the quality of the firework's objective function value. A Gaussian operator generates sparks to increase solution diversity, and a selection mechanism is applied to determine which particles (fireworks, explosion sparks, and Gaussian sparks) move on to the next generation. As outlined by Tan & Zhu (2010) and Pang et al. (2020), the number and amplitude of sparks are key elements in the algorithm's exploration process. Superior fireworks are closer to the best solution in the population, and the number of sparks is carefully controlled to prevent premature convergence. At each iteration, the best solution is retained, and n_fire-1 particles are selected based on their relative positions to promote diversity. Less crowded regions in the solution space are favored, increasing the likelihood of those solutions being selected for the next generation. Unlike other population-based metaheuristics, FWA uniquely considers multiple types of individuals (fireworks, explosion sparks, Gaussian sparks) generated by two distinct probability distributions (uniform and Gaussian). Moreover, FWA doesn't simply transfer the best solutions to the next generation but probabilistically chooses from less populated regions, promoting greater population diversity. Therefore, the leading contributions of this article lie in the study of a realistic scheduling problem which has been relatively little studied in the literature, namely in applying a comparatively new meta-heuristic which has not yet been experimented with in the problem addressed herein, and which has obtained good results in other optimization problems. Furthermore, in proposing a parameter tuning method based on the impact of the combination of values, and in the use of a modern, scientific, and efficient programming language.

The remaining paper is organized through the following sections: the Fireworks Algorithm and some application studies are presented in Section 2. Section 3 provides an elaborated description of the problem, and the proposed metaheuristic is introduced. Section 4 describes the experiment results, comparative study, and managerial implications of the study. Finally, Section 5 concludes the paper.

2. Problem description and general notations

Let $J = \{J_1, J_2, \ldots, J_n\}$ be the set of n jobs that have to be scheduled on a unique machine. Each job J_j has processing time p_j, an associated time window $[e_j, d_j]$, where e_j is the earliest due date and d_j is the most delayed due date. If job J_j is completed before e_j, then there is a cost of w_j^E per unit of earliness time. When the job J_j is completed after d_j, there is a cost w_j^T per unit of tardiness time. Jobs finished within their due windows do not lead costs. The machine can only perform a single job at a time and it cannot be interrupted once the process is initiated. All jobs are available at the beginning of the processing. Between two consecutive jobs J_i and J_j, a setup time of s_{ij} is required. The setup time for the machine to process the first job in the sequence is assumed to be zero. Allowing idle time between the execution of consecutive jobs is acceptable. The completion time of job J_j is represented by C_j, whereas the earliness and tardiness times of J_j are represented by $E_j = \max\{0, e_j - C_j\}$ and $T_j = \max\{0, C_j - d_j\}$, respectively. The goal is to establish a job sequence π and corresponding start times that minimize the weighted sum of each job's earliness and tardiness, i.e. minimize the value of $ET_w = \sum_{j=1}^{n}(w_j^E E_j + w_j^T T_j)$. Tables 1 and 2 present the notation of the addressed problem and the notation of the proposed FWA, respectively.

Table 1
Notation of the problem $1|<e_j, d_j>, s_{ij}|ET_w$

n	number of jobs
J_j	job J_j, $j = 1, \ldots, n$
J	set of jobs
p_j	processing time of job J_j
e_j	earliest due date of job J_j
d_j	tardiest due date of job J_j
s_{ij}	sequence-dependent setup time between job Ji and J_j
C_j	completion time of job J_j
E_j	earliness of job J_j
T_j	tardiness of job J_j
w_j^E	weight of the earliness of job J_j
w_j^T	weight of the tardiness of job J_j
ET_w	weighted sum of the earliness and tardiness (objective function)

2. Algorithms evaluated

Table 2
Notation of FWA

i	Index of fireworks	
l	index of sparks	
j	index of dimensions of the locations (fireworks and sparks)	
L	set of all current locations (fireworks and sparks)	(8), (9)
n_fire	number of fireworks in the population	(1), (2)
x_i	firework or solution i, $i = 1, \ldots, n\_fire$	(1), (2)
$f(x_i)$	fitness value of x_i	(1), (2)
x_i'	general location (fireworks, explosion sparks or gaussian sparks)	(8), (9)
x_{ik}	firework i with dimension k	(4), (6)
x_{lk}^e	explosion spark l with dimension k	(4), (5)
x_{lk}^g	gaussian spark l with dimension k	(6), (7)
x_{min}	lower bound of the search space	(5), (7)
x_{max}	upper bound of the search space	(5), (7)
y_{min}	minimum (best) values of $f(x_i)$	(1), (2)
y_{max}	maximum values of $f(x_i)$	(1), (2)
A_i	explosion amplitude the firework i	(1)
A_{max}	parameter of the maximum range of the fireworks explosion	(1)
S_i	number of explosion sparks of the firework i	(2), (3)
m_spark	parameter value of the quantity of the explosion sparks	(2), (3)
g_spark	parameter value of the quantity of the gaussian sparks	
a	parameter which confines the range of the number of sparks	(3)
b	parameter which confines the range of the number of sparks	(3)
z	number of dimensions selected	
g	coefficient of gaussian explosion, $g = N(1, 1)$	(6)
$R(x_i')$	general distance between a location x_i' and other locations	(8), (9)
$d(x_i', x_j')$	distance between the location x_i' and x_j'	(9)
$p(x_i')$	selection probability of a location x_i'	(8)
ε	the smallest constant to avoid zero-division error	(1), (2)
$rand$	random number	(4)
$round$	rounding function	(3)
mod	modular function	(5), (7)

We kept our FWA continuous to preserve the original analogy in the explosions and the mutation. Therefore, we adopted the Smallest Position Value (SPV) rule to convert the real values of the locations of fireworks and sparks into the permutational solution of the job scheduling problem. The pseudocode of the proposed Firework is described in Algorithm 1 as follows.

Algorithm 1: Algorithm FWA

```
Input: $p, e, d, w^T, w^E, w^T$
Output: $\pi$, ETw
procedure FWA
Initialize n_fire, m_spark, g_spark, a, b, $A_{max}$, termination
Set $x_{min} = -100$, $x_{max} = 100$, and $\xi = 10^{-38}$
Generate randomly n_fire fireworks and calculate their fitness
Save the best ($y_{min}$) and worst ($y_{max}$) values
while termination not reached do
        for $i = 1$:n_fire do
                Calculate explosion amplitude:
                        $A_i = A_{max} \frac{f(x_i) - y_{min} + \varepsilon}{\sum_{i=i}^{n\_fire} (f(x_i - y_{min}) + \varepsilon}$        (1)
                where   $A_{max}$: maximum range of the fireworks explosion,
                        $x_i$: firework $i$,
                        $f(x_i)$: fitness value of $x_i$,
                        $y_{min} = min\, f(x_i)$, $i = 1, \ldots, n\_fire$,
                        n_fire: number of fireworks in the population,
                        $\varepsilon$: constant to avoid a zero-division error.

                Calculate the number of sparks:
                        $S_i = m\_spark \frac{y_{max} - f(x_i) + \varepsilon}{\sum_{i=i}^{n\_fire} (f(y_{max} - x_i) + \varepsilon}$        (2)
                where   m_spark: parameter value to restrict the quantity of the sparks,
                        $y_{max} = max\, f(x_i)$, $i = 1, \ldots, n\_fire$.

                Bound the number of explosion sparks:
                        $S_i = \begin{cases} round(a * m\_spark), \text{if } S_i < a * m\_spark \\ round(b * m\_spark), \text{if } S_i > b * m\_spark, a < b < 1 \\ round(S_i), \text{otherwise} \end{cases}$        (3)
                where $a$ and $b$: parameters ranging the sparks number,
                round: rounding function.

                Generate the explosion sparks:
                        $x_{lk}^e = x_{ik} + A_i\, rand(-1,1)$        (4)
                where rand(–1,1): random number with distribution $U[0,1]$.

                Map not exceeding the search space limits:
                        $x_{lk}^e = x_{min} + |x_{lk}^e|\, mod(x_{max} - x_{min})$        (5)
                where mod: modular function,
                $x_{min}$ and $x_{max}$: bounds of the solution space.
        end for
        for $l = 1$:g_spark do
                Select randomly a firework $i$
                Generate the gaussian sparks:
                        $x_{lk}^g = x_{ik}\, g$        (6)
                where $g$: coefficient of the gaussian explosion, $g = N(1, 1)$.

                Map to not exceed the boundary of the search space:
                        $x_{lk}^g = x_{min} + |x_{lk}^g|\, mod(x_{max} - x_{min})$.        (7)

        end for
        Select the best location (among fireworks and sparks) and keep it for the next ration
        Select n_fire–1 particles $x_i'$ for the next generation with probability $p(x_i')$:
                        $p(x_i') = \frac{R(x_i')}{\sum_{j \in L} R(x_j')}$.        (8)

        where
                        $R(x_i') = \sum_{j \in L} d(x_i', x_j') = \sum_{j \in L} \|x_i' - x_j'\|$        (9)
        is the distance between a location $x_i'$ and other locations and $L$ is set of all current locations of fireworks and sparks.
end while
return best individual and fitness
end procedure
```

Our FWA is similar to Tan & Zhu (2010). The special differences lie in the conversion rule to the permutational solution and the values of the calibrated parameters. In addition to applying the relatively new metaheuristic in the scheduling problem, the proposal of the tuning method, which is another contribution of this work, will be detailed in the next section. For each sequence generated by the FWA, it is necessary to call a job positioning algorithm to determine the optimal date for each job to be performed. For this purpose, we employed in this paper the Idle Time Insertion Algorithm (ITIA) of Rosa et al. (2017). ITIA is a $O(n^2)$ algorithm designed to determine the start time for each job in a given sequence, providing the optimal allocation of idle times.

4. Computational experiments

For a fair comparison, all the algorithms were coded in Julia Language version 1.36.0 and run on the same computer with the following configuration: Intel(R) Core(TM) i5-8265U processor with 1.80GHz, 16GB of RAM memory, and one SSD.
We used the benchmark instances of Rosa et al. (2017), composed by instances with 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 30, 40, 50, and 75 jobs. According to Rosa et al. (2017), for each job J_j, the processing times p_j, setup times s_{ij}, cost per tardiness unit w_j^T, and cost per earliness unit w_j^E were randomly generated integers within the intervals [1, 40], [5, 15], [1, 10], and [1, w_j^T], respectively. The time window center of job J_j was a random integer number within the interval $[(1 - TF - RDD/2)TPF, (1 - TF + RDD/2)TPT]$, where TPT is the sum of processing times of all jobs, TF is the tardiness factor, and RDD is the relative range of the time windows. Therefore, the time window width is an integer number randomly selected in the interval $[0, TPT/n]$ (n is the number of jobs to be scheduled). The values 0.1, 0.3, 0.5, and 0.8 were used for TF, and 0.4, 0.7, 1.0, and 1.3 for RDD. Thus, each set has 16 instances, totaling 272 instances.
Next, five independent runs were carried out for each algorithm for each instance. We calculated the average relative percentage deviation from the best-known solution for each instance. The stopping criteria for each heuristic was $5*n$ iterations, with n equal the number of jobs.

4.1 Proposed parameter tuning

We developed a parameter tuning procedure inspired by the idea of Joshi & Bansal (2020). They presented a novel approach based on the relation between the algorithm's performance and functional landscape and evaluated it in the Gravitational Search Algorithm (GSA) with a test suite of continuous functions. The rationale behind the procedure is to find the most influential values of any parameter combination. Let *EW* be a matrix of "elementary weights" with the first dimension being the *popsize* and the others corresponding to the number of parameters to be calibrated. So the total number of dimensions of matrix *EW* is the number of parameters plus 1. For example, if a metaheuristic has 3 parameters to be calibrated, the matrix *EW* will have 4 dimensions, one for the population and one for each of the parameters.
Also, let *Fitness1* and *Fitness2* be matrices of objective function values obtained by all combinations of parameters. Matrices *EW* and *Fitness1* are initialized with all values equal to 1.

Algorithm 2: Parameter tuning

```
Input: instance, parameters, range_of_parameters
Output: best combination of parameters
procedure parameter_tuning
Initialize termination, EW[] = 1, Fitness1[] = 1, bigger_EW = –100
Calculate the objective function Fitness2[] for each parameter combination
while termination not reached do
        for each parameter combination
                EW[] = abs((Fitness2[] – Fitness1[])/popsize)
        end for
        for each parameter combination α
                if mean(EW[α]) > bigger_EW
                        bigger_EW = mean(EW[α])
                        save the combination α
                end if
        end for
        for each parameter combination
                Fitness1[] = Fitness2[]
        end for
        for each parameter combination
                EW[] = abs((Fitness2[] – Fitness1[])/popsize)
        end for
end while
return best combination α of parameters
end procedure
```

As demonstrated in the pseudocode of Algorithm 2, the proposed procedure for parameter tuning assesses the impact of the influence of changing the parameter combination, differently from most of the other procedures which evaluate the performance of each parameter value individually.

4.2 Comparison to the literature

The results of the proposed FWA were compared to the Iterated Greedy Algorithm (IGA) of Pan & Ruiz (2014), considered the state-of-the-art heuristic for different scheduling problems, and three other metaheuristics: Artificial Bee Colony (ABC) and Bat Algorithm (BA) of the work of Agarwal & Mehta (2018) and Particle Swarm Optimization (PSO) adapted from Tasgetiren et al. (2007). All these metaheuristics were calibrated by the proposed procedure specifically for the addressed problem. Consequently, the parameters for these heuristics found by the tuning are proposed as follows: for ABC, the threshold value for entering into the scout bee phase is *limit* = 81, for BA, the loudness sound $A = 0.1$ and the pulse emission rate $r = 0.1$, for PSO, the inertia weight $w = 0.35$ and the acceleration coefficients $c = 0.85$, for IGA, the parameter for adjustment of the temperature $\lambda =$ and the destruction size $d = 4$, and for the FWA, *m_spark* = 5, *g_spark* = 1, $a = 0.2$, $b = 0.2$ and $A_{max} = 35$. We calculated the relative percentage deviation (RPD) from a reference solution as a response variable for the experiments as follows:

$$RPD_h = \frac{(ET_{w,h} - ET_w^*)}{ET_w^*} \tag{10}$$

where $ET_{w,h}$ is the weighted sum of the earliness and tardiness generated in the instance h by a given algorithm, and ET_w^* is the best-weighted sum of the earliness and tardiness found by any of the algorithms. After executing the metaheuristics and collecting all the outputs, results of the experiment were analyzed by means of the Analysis of Variance (ANOVA) technique. The means plots with 95% Tukey's test confidence intervals are given in Fig. 1 and Fig. 2. Overlapping confidence intervals means that the observed difference in the response variable (RPD) of the two overlapped means is statistically insignificant. By Fig. 1, the superior result of FWA is evident.

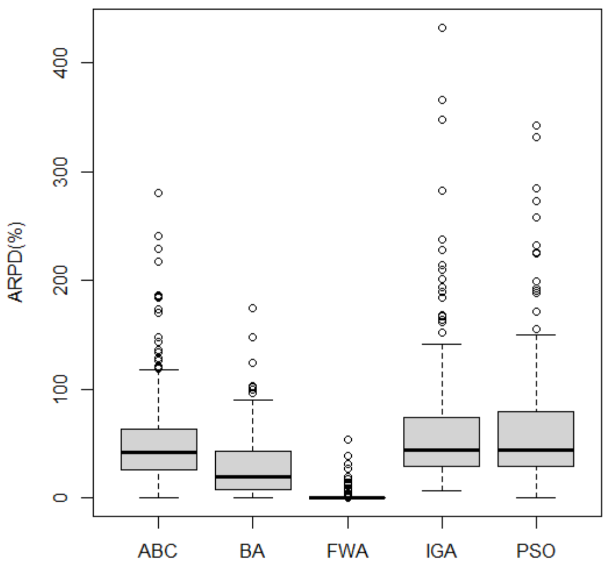

Fig. 1. Means plot of average RPD (%) for Tukey's test with 95% confidence intervals

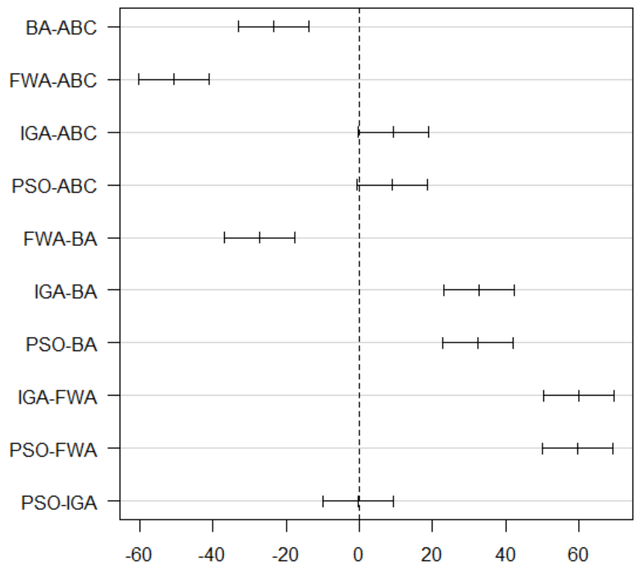

Fig. 2. Statistical comparison among algorithms with Tukey's test with 95% confidence intervals

It can be seen in Fig. 2 that there is no statistical difference between the results of IGA and ABC, PSO and ABC, and PSO and IGA. Oppositely, there exists a statistically significant difference between FWA and any other algorithm. Further, a detailed performance comparison in terms of average RPD is summarized in Table 3, grouped for each dimension of the instances which refers to the number of jobs of the instance class.

As can be noticed in Table 1, the FWA has the lowest average RPD, or the relative error, in every dimension of the instances, which indicates its better results overall. Furthermore, the FWA heuristic achieved the best solution in 9 of 17 dimensions, i.e., a relative deviation of zero. The RPD was almost zero in several others. Thus, the FWA outperformed the other four implemented algorithms of the literature in all the classes of instances solved. We can visually compare these results in Figure 3 with the average RPD of the five metaheuristics by each dimension.

Table 3
Computational results (in average RPD) of the algorithms for each instance dimensions

Dimension	ABC	BA	PSO	IGA	FWA
8	49.90	12.28	63.99	63.10	3.28
9	49.71	14.51	64.76	62.21	7.76
10	61.40	28.20	82.94	89.04	0.66
11	61.65	26.13	69.68	75.04	6.16
12	47.52	20.48	55.71	62.66	1.59
13	54.11	24.20	56.42	57.68	0.90
14	70.91	25.91	72.32	72.09	0.00
15	54.08	30.68	66.08	66.72	0.00
16	48.94	31.31	66.13	59.94	0.00
17	56.18	31.76	66.02	59.41	0.26
18	51.58	28.43	58.22	53.77	0.00
19	48.41	28.17	46.28	49.23	0.23
20	51.93	34.71	55.70	56.43	0.00
30	45.95	36.20	57.06	58.74	0.00
40	49.24	42.82	63.79	59.70	0.00
50	42.18	34.60	48.65	48.25	0.00
75	38.62	33.29	42.34	46.09	0.00
Average	**51.90**	**28.45**	**60.95**	**61.18**	**1.23**

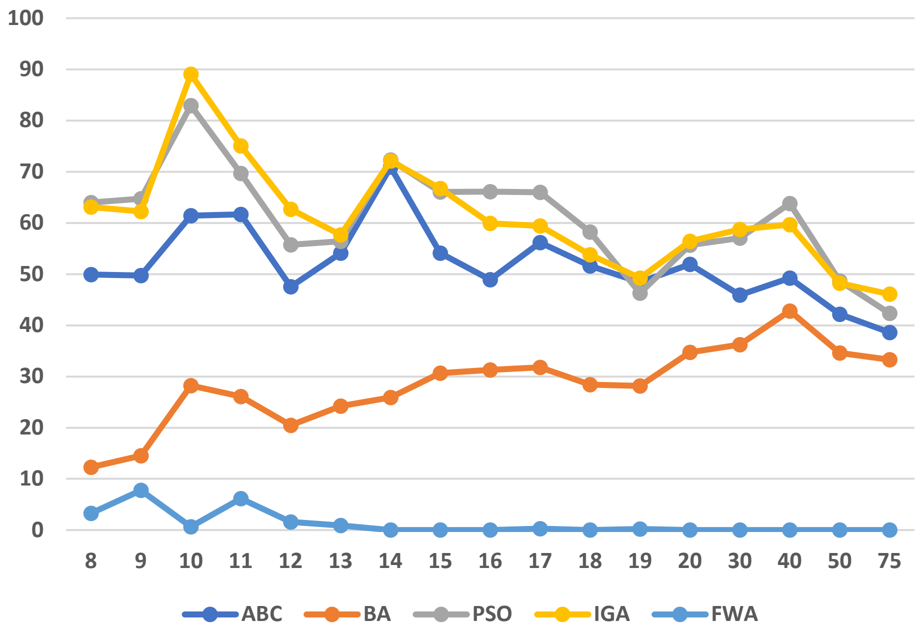

Fig. 3. RPD (%) of the algorithms by dimension

The instability of the deviations of the three worst algorithms (ABC, PSO, and IGA) is shown quite clearly in Figure 3, especially for instances with smaller dimensions, and the crescent values of BA, which is in the second-best results but significantly far from the best. The smaller deviations of FWA can be observed, with practically zero value from dimension 14 onwards to the largest instances. The algorithms were compared using the performance profiles proposed by Dolan & Moré (2002) to better highlight the results, a very used tool for benchmarking and evaluating the performance of several software programs when run on a testbed set. As pointed out by Gould & Scott (2016), performance profiles provide a very useful and convenient means of assessing the performance of a solver relative to the best solver on each instance from that set. Moreover, this resource offers an estimate of the expected performance difference between the solvers. Several recent and noteworthy works on various scheduling problems have successfully leveraged this method on its evaluation process are Abreu & Fuchigami (2022), Freitas & Fuchigami (2022), and Moreno et al. (2019). Let T represent a set of n_T instances and S a set of n_S algorithms. Suppose a given algorithm $i \in S$ reports a statistic $s_{ij} \geq 0$ when runs an instance $j \in T$, and the smaller this statistic, the more effective the model is. The *performance ratio* is defined as $r_{ij} = s_{ij}/\hat{s}_j$. where $\hat{s}_j = \min\{s_{ij}: i \in S\}$. For each $i \in S$ and $\tau \geq 1$, we define a $k(r_{ij}, \tau)$ as follows.

$$k(r_{ij}, \tau) = \begin{cases} 1 \; if \; r_{ij} \leq \tau \\ 0 \; otherwise \end{cases} \tag{11}$$

The performance profile of the algorithm i is then given by the following function (11) with $\tau \geq 1$.

$$P_i(\tau) = \frac{\sum_{j \in T} k(r_{ij}, \tau)}{n_T} \tag{12}$$

The performance profile of a method is represented by its cumulative distribution function, which shows the probability that the performance ratio remains below a specified threshold $(r_{ij} \leq \tau)$. Consequently, $P(\tau)$ reflects the percentage of instances where the method obtains a result within τ times the optimal value of the measure. Fig. 4 presents this performance profile comparison.

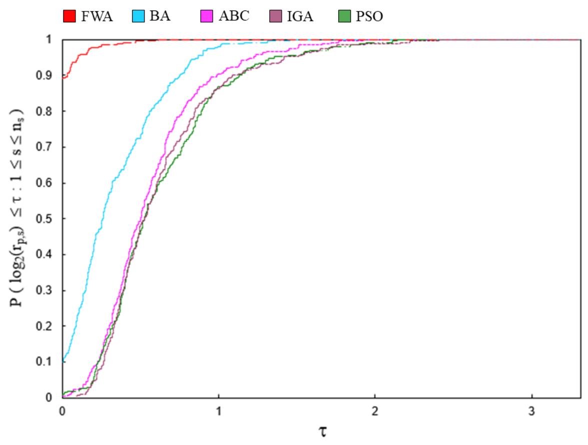

Fig. 4. Performance profiles of the algorithms on the objective values

According to Fig. 4, performance profiles visually demonstrate how the FWA outperformed the other algorithms. The red line starting at about 0.9 indicates that the FWA results were better in almost 90% of the instances. Also, FWA is the fastest to achieve 100% of the best results. Figure 2 still endorses the previous analysis, showing BA as the second-best algorithm and the other three (ABC, IGA, and PSO) have curves with similar behaviors. Concerning the execution times, Table 4 presents the average CPU times, in seconds, of each metaheuristic.

Table 4
Average CPU times (in seconds) of each algorithm

ABC	**BA**	**PSO**	**IGA**	**FWA**
0.1483	0.1602	0.1554	0.1622	0.1551

As can be seen in Table 4, the five algorithms proved to have acceptable execution times, with similar values of CPU times, and all of them spent around 0.1 second on average.1

5. Final considerations

In this study we have addressed a just-in-time scheduling problem with distinct time windows and sequence-dependent setup times with minimization of total weighted earliness and tardiness of jobs, achieving the proposed goal to implement an efficient metaheuristic not yet employed in this problem. Our proposed Fireworks Algorithm outperformed state-of-the-art metaheuristics for scheduling problems, especially the IGA. We approached a realistic scheduling problem little studied in the specific literature, and we implemented five different metaheuristics employing a data set benchmark of the literature to compare the performance of the algorithms. Also, we proposed a parameter tuning method based on the impact of combining values. The computational results showed that the FWA outperformed the other algorithms in all of the problem dimensions compared and achieved the absolute best result (100% of success i.e., relative percentage deviation of zero) in 9 of the 17 instance classes analyzed, especially in the large-sized instance. Thus, it highlights the applicability and efficiency of the FWA in large-sized real problems. For future research, we suggest the considerations of different restrictions in the problem, for example, release times and/or specific weights for each job; in the FWA, could be investigated diverse procedures for search in the solution space and, being an atypical metaheuristic in the sense of considering two probability distributions (uniform and gaussian) and a specific probabilistic model (for the particles choice for the next generation), other theoretical probability distributions could be tested. Also, the FWA algorithm can be adapted for other scheduling problems, given its good performance in the problems already applied.

Funding details

This work was supported by the Sao Paulo Research Foundation for financial support (FAPESP, grant numbers 2022/05803-3, 2022/10993-6, and 2023/08678-8) and the Brazilian National Council for Scientific and Technological Development (CNPq, grant numbers PI01143-2015/7, PI01143-2015/11, 154540/2019-6, 135179/2020-3, 143568/2023-0, and 144305/2024-0).

References

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

Agarwal, P., & Mehta, S. (2018). Empirical analysis of five nature-inspired algorithms on real parameter optimization problems. *Artificial Intelligence Review, 50*, 383-439.

Ahmadizar, F., & Farhadi, S. (2015). Single-machine batch delivery scheduling with job release dates, due windows and earliness, tardiness, holding and delivery costs. *Computers & Operations Research, 53*, 194-205.

Alidaee, B., Li, H., Wang, H., & Womer, K. (2021). Integer programming formulations in sequencing with total earliness and tardiness penalties, arbitrary due dates, and no idle time: a concise review and extension. *Omega, 103*, 102446.

Allahverdi, A. (2015). The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research, 246*(2), 345-378.

Allahverdi, A., & Soroush, H.M. (2008). The significance of reducing setup times/setup costs. *European Journal of Operational Research, 187*(3), 978-984.

Allahverdi, A., Gupta, J.N., & Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *Omega, 27*(2), 219-239.

Baker, K.R., & Scudder, G.D. (1990). Sequencing with earliness and tardiness penalties: a review. *Operations Research, 38*, 22-36.

Ding, R., Hu, S., Xing, Z., & Yan, T. (2025). Multi-type radar deployment for UAV swarms defense coverage using Firework Algorithm with Determinantal Point Processes under complex terrain. *Applied Soft Computing, 170*, 112681.

Dolan, E. D., & Moré, J.J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming, 91*(2), 201-213.

Freitas, M.G., Fuchigami, H.Y. (2022). A new technology implementation via mathematical modeling for the sequence-dependent setup times of industrial problems. *Computers & Industrial Engineering, 172*, 108624.

Fuchigami, H. Y., & Rangel, S. (2018). A survey of case studies in production scheduling: Analysis and perspectives. *Journal of Computational Science, 25*, 425-436.

Fuchigami, H. Y., Sarker, R., & Rangel, S. (2018). Near-optimal heuristics for just-in-time jobs maximization in flow shop scheduling. *Algorithms, 11*(4), 43.

Fuchigami, H.Y., Moccellin, J.V., & Ruiz, R. (2015). New priority rules for the flexible flow line scheduling problem with setup times. *Production, 25*, 779-790.

Gould, N., & Scott, J. (2016). A note on performance profiles for benchmarking software. *ACM Transactions on Mathematical Software (TOMS), 43*(2), 1-5.

Hao, T., Ma, Z., & Wang, Y. (2024). An enhanced fireworks algorithm and its application in fault detection of the displacement sensor. *Measurement: Sensors, 34,* 101250.

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*, 100575.

Janiak, A., Janiak, W.A., Krysiak, T., & Kwiatkowski, T. (2015). A survey on scheduling problems with due windows. *European Journal of Operational Research, 242*, 347-357.

Joshi, S.K., & Bansal, J.C. (2020). Parameter tuning for meta-heuristics. *Knowledge-Based Systems, 189*, 105094.

Józefowska, J. (2007). *Just-in-Time Scheduling: Models and Algorithms for Computer and Manufacturing Systems*. Springer Science: New York, NY, USA.

Jula, P., & Kones, I. (2013). Continuous-time for scheduling a single machine with sequence-dependent setup times and time window constraints in coordinated chains. *International Journal of Production Research, 51*, 3654-3670.

Khadivi, M., Charter, T., Yaghoubi, M., Jalayer, M., Ahang, M., Shojaeinasab, A., & Najjaran, H. (2025). Deep reinforcement learning for machine scheduling: Methodology, the state-of-the-art, and future directions. *Computers & Industrial Engineering, 200*, 110856.

Koulamas, C. (1996). Single-machine scheduling with time windows and earliness/tardiness penalties. *European Journal of Operational Research, 91*, 190-202.

Li, J., & Tan, Y. (2019). A comprehensive review of the fireworks algorithm. *ACM Computing Surveys, 52*, 1-28.

Li, X.-G., Han, S.-F., & Gong, C.-Q. (2017). Analysis and improvement of fireworks algorithm. *Algorithms, 10*, 1-22.

Liu, X., & Qin, X. (2021). A neighborhood information utilization fireworks algorithm and its application to traffic flow prediction. *Expert Systems with Applications, 183*, 115189.

Meng, X., & Tan, Y. (2024). Multi-guiding spark fireworks algorithm: Solving multimodal functions by multiple guiding sparks in fireworks algorithm. *Swarm and Evolutionary Computation, 85*, 101458.

Moreno, A., Munari, P., & Alem, D. A branch-and-benders-cut algorithm for the crew scheduling and routing problem in road restoration. *European Journal of Operational Research, 275*(1), 16-34, 2019.

Pan, Q.-K., & Ruiz, R. (2014). An effective iterated greedy algorithm for the mixed no-idle permutation flowshop scheduling problem. *Omega, 44*, 41-50.

Pang, X., Xue, H., Tseng, M.-L., Lim, M.K., & Liu, K. (2020). Hybrid flow shop scheduling problems using improved fireworks algorithms for permutation. *Applied Sciences, 10*, 1-16.

Penna, P.H.V., Souza, M.J.F., Gonçalves, F.A.C.A., & Ochi, L.S. (2012). Uma heurística híbrida para minimizar custos com antecipação e atraso do sequenciamento da produção em uma máquina. *Produção, 22,* 766-777.

Ribeiro, F.F., Souza, M.J.F., & Souza, S.R. (2010). An adaptive genetic algorithm to the single machine scheduling problem with earliness and tardiness penalties. *Lecture Notes in Artificial Intelligence, 6403*, 203-212.

Ríos-Solís, Y.A., & Ríos-Mercado, R.Z. (2012). *Just-In-Time Systems*. Springer Sciences: New York, NY, USA.

Rosa, B.F., Souza, M.J.F., & Souza, S.R. (2018). Algorithms based on VNS for solving the single machine scheduling problem with earliness and tardiness penalties. *Electronic Notes in Discrete Mathematics, 66*, 47-54.

Rosa, B.F., Souza, M.J.F., Souza, S.R., França Filho, M.F.F., Ales, Z., & Michelon, P.Y. (2017). Algorithms for job scheduling problems with distinct time windows and general earliness/tardiness penalties. *Computers and Operations Research, 81*, 203-215.

Shabtay, D., & Steiner, G. (2012). *Scheduling to maximize the number of just-in-time jobs: A survey*. In Just-in-Time Systems, Ríos-Solís, Y.A., Ríos-Mercado, R.Z., Eds. Springer Sciences: New York, NY, USA.

Tan, Y. (2015). *Fireworks Algorithm: a novel swarm intelligence optimization method*. Springer.

Tan, Y., & Zhu, Y. (2010). Fireworks algorithm for optimization. *Proceedings of the International Conference in Swarm Intelligence. Springer*, 355-364.

Tasgetiren, M.F., Liang, Y.-C., Sevkli, M., & Gencyilmaz, G. (2007). A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. *European Journal of Operational Research, 177*, 1930-1947.

Xu, Z., Zhang, K., Xu, X., & He, J. (2020). A fireworks algorithm based on transfer spark for evolutionary multitasking. *Frontiers in Neurorobotics, 13*, 1-14.

Ye, M., Bartolini, E., & Schneider, M. (2024). A general variable neighborhood search for the traveling salesman problem with time windows under various objectives. *Discrete Applied Mathematics, 346*, 95-114.

Zhao, C., & Tang, H. (2010). Single machine scheduling with past-sequence-dependent setup times and deteriorating jobs. *Computers & Industrial Engineering, 59*, 663-666.