Vol.67: e24240084, 2024
https://doi.org/10.1590/1678-4324-2024240084
ISSN 1678-4324 Online Edition

Article - Engineering, Technology and Techniques

Innovative Optimization Algorithms for Large-Sized Industrial Scheduling Problems

Helio Yochihiro Fuchigami[1*]
https://orcid.org/ 0000-0002-8943-1506

Alex Paranahyba Abreu[1]
https://orcid.org/ 0000-0002-8338-7452

[1]Universidade Federal de São Carlos (UFSCar), Departamento de Engenharia de Produção, São Carlos, São Paulo, Brasil.

Editor-in-Chief: Alexandre Rasi Aoki
Associate Editor: Fabio Alessandro Guerra

Received: 19-Feb-2024; Accepted: 13-May-2024

*Correspondence: helio@dep.ufscar.br; Tel.: +55-16-3351-9510 (H.Y.F.).

HIGHLIGHTS

- Two new coronavirus-based metaheuristic algorithm was proposed.
- A challenging nonlinear optimization problem is solved efficiently.
- A mathematical optimization model is formulated to solve small instances.
- Our algorithms outperform the main metaheuristics from the literature.

Abstract: The problem of minimizing total quadratic completion time in flow shops presents a significant challenge in industries such as chemical, metallurgical, and ceramic manufacturing. Initially investigated by Ren and coauthors (2016) [1], this problem addresses the need to balance intermediate inventory reduction with maximizing resource utilization, particularly in multi-objective scenarios. We proposed two innovative metaheuristics (Covid and CHIO algorithms) and a mathematical optimization model. Evaluations were conducted across two industrial settings and three additional benchmarks from existing literature. Through the statistical analysis and performance profiling, our findings indicate that the Covid and CHIO algorithms outperform the Differential Evolution of Ren and coauthors (2016) [1] and the Iterated Greedy Algorithm of Pan and Ruiz (2012) [2], as well as other techniques. Notably, the proposed Covid and CHIO algorithms achieved an average relative deviation from the optimal solution of 0.41% and 0.23%, respectively. Furthermore, they consistently outperformed other methods, securing the best solution in at least 12.5% of instances across all benchmarks, with their worst solutions closer to the best solutions than those produced by alternative approaches.

Keywords: flow shop; metaheuristics; mathematical optimization; total quadratic completion time; stochastic local search.

INTRODUCTION

Flow shop scheduling is a frequent optimization problem in industrial production due to its practical and conceptual importance. In a permutation environment, the sequence of jobs, once defined, must be processed in the same order by each machine. Therefore, the classical structure of the permutation flow shop scheduling problem is characterized as a system of machines in which the flow of jobs is unidirectional and linear, i.e., the machines are arranged in series and the same sequence of tasks must be processed by the machines [3, 4].

Many performance measures were considered in this environment, especially the traditional makespan and flowtime [5], followed by earliness and tardiness and their related functions [6], as can be seen in comprehensive reviews about the classic flow shop [7-10]. In this article, we consider a more complex flow shop problem with a nonlinear objective function, the total quadratic completion time (TQCT). According to the standard scheduling three-field notation [11], this problem can be represented by $Fm||\sum C_j^2$, where *Fm* is the flow shop production environment with a general number of *m* machines and C_j denotes the completion time of each job *j* in the last machine. Other variations of this problem are the total *k*-power completion time (TKCT) and the total weighted quadratic completion time (TWQCT).

Quadratic criteria are suitable for the multi-objective industrial setting [1]. For example, in elevated temperature material handling processes, the waiting times of intermediate products will consume a significant amount of energy. Therefore, optimizing the TQCT objective may be an effective way to reduce energy consumption in environments like steel, aluminum, zinc, brick and tile, and chemical industries.

A practical application of the TQCT criteria can be found in high-energy-consuming environments. In the steel-making industries, the rolling process requires that the slab achieves a determined elevated temperature (after the heating furnace) to be processed by the rolling mill. A prohibitive waiting time can overly cool a pre-heated slab. If the temperature of a slab drops below a limit value, it is necessary to reheat it, overconsuming the energy. There is a nonlinear relation between the waiting time and the temperature drop. So, the later the slab is rolled, the greater the energy loss per unit of waiting time. In this case, the TQCT measure can dramatically shorten the waiting time for heat slabs and the consumption of energy for such industries [12, 13].

In industrial applications, makespan decreases machine loads and economizes energy consumption, whereas total completion time reduces work-in-process and saves inventory costs. Cheng and Liu (2004) [14] stated that quadratic cost functions are more appropriate, compared with the linear one, for cases where the later a job is finished, the greater the cost per unit of elapsed time. They also pointed out that using the TQCT as the objective function can be considered a trade-off between the makespan and the total completion time.

Among all the diverse flow shop problems, few researchers investigated the $Fm||\sum C_j^2$ and this is one of our motivations in the present work. As Koulamas and Kyparisis (2005) [15] proved that this problem is strongly NP-hard, heuristics methods are more appropriate to solve it, especially in large-size cases. In this research, nine solution methods were computationally implemented and evaluated. For the addressed problem, we propose two coronavirus optimization algorithms: Covid and CHIO (Coronavirus Herd Immunity Optimization Algorithm). Also, we applied other metaheuristics for comparison under the following justifications: Fireworks Algorithm (FWA) is non-nature inspired with a peculiar way of generating the populations and has shown good results in different optimization problems; Particle Swarm Optimization (PSO) is a classic swarm intelligence algorithm; and Jaya and Teaching-Learning Based Optimization (TLBO) are relatively new and do not require algorithm-specific parameters, just the controlling parameter of population size.

Finally, we compared the performances of the assessed metaheuristics with the optimal (or best found) solution of a mathematical optimization model, the Iterated Greedy Algorithm (IGA), known as the state-of-the-art algorithm for flow shop scheduling problems, and the Differential Evolution (DE) algorithm of Ren and coauthors (2016) [1] applied for the same problem.

The remainder of the article is organized as follows: the related problems and recent applications are presented in Section 2. In Section 3, the problem is mathematically formulated. Section 4 introduces the proposed solution methods. Section 5 discusses the results and section Conclusion completes the work.

RELATED LITERATURE

Despite the single machine problem with TQCT having been studied for the last decades, as pointed out by the literature review of Bai and coauthors (2020) [16], this objective function has been little approached in the flow shop problem. Koulamas and Kyparisis (2005) [15] seem to be the first to address the flow shop problem with TQCT. They studied the worst-case performance bounds for the TQCT and other measures in the flow shop problem.

The generalized total *k*-power completion time in flow shop and open shop was investigated by Bai and Zhang (2014) [13] which presented lower bounds and a study of the worst-case of the problem. For the dynamic TKCT flow shop with release dates, Bai and coauthors (2017) [17] proposed a branch-and-bound algorithm for small-scale instances (up to 15 jobs and 8 machines) and a discrete Differential Evolution algorithm for medium-scale instances (up to 180 jobs and 10 machines).

Ren and coauthors (2012) [18] presented an improvement strategy with local search to promote the performance of the classical shortest processing time (SPT) heuristic in the TQCT flow shop problem. Bai (2015) [12] proved the asymptotic optimality of two SPT-based heuristics for the TQCT flow shop with release dates. And Ren and coauthors (2016) [1] proposed a discrete Differential Evolution algorithm for minimizing the total weighted quadratic completion time in the flow shop.

Several researchers considered the learning effect constraint in the TQCT flow shop problem. In the learning effect case, the processing times of jobs are defined as functions of their positions in a permutation, in general, the later the shorter. Some works set out to analyze worst-case bounds for the TQCT flow shop problem with learning effects [19-23]. Bai and coauthors (2018) [24] proposed a branch-and-bound algorithm, a mixed-integer programming model, and proved the asymptotic optimality of a class of shortest processing time available (SPTA)-based heuristics for TQCT flow shop with learning effect and release dates.

As can be observed in this literature review, the flow shop problem with TQCT has been little investigated, and most research has taken a theoretical approach to the problem, with no computational experimentation. This fact highlights a research gap showing the importance of this research. Thereby due to the complexity of the problem $Fm||\sum C_j^2$, which is strongly NP-hard as mentioned, we have chosen the metaheuristic techniques to solve the problem, especially the large-size instances.

Metaheuristics deals with large and complex problems providing near-optimal solutions within a reasonable time. In the review ‘Metaheuristics “in the large”’, Swan and coauthors (2022) [25] defended extensible and re-usable algorithm templates, white box problem descriptions that provide generic support for the injection of domain-specific knowledge, and remotely accessible frameworks, components, and problems. Zhao and Kong (2016) [26] proposed an improved genetic algorithm for the mixed flow shop problem.

Recently, inspired by the pandemic of Covid-19, new metaheuristics were proposed motivated by the power of spreading behavior of the virus. Martínez-Álvarez and coauthors (2020) [27] introduced a Coronavirus Optimization Algorithm (CVOA) based on the concepts of social distancing measures, super-spreading rate, reinfection probability, and traveling rate to solve the problem of electricity load time series forecasting. Hosseini and coauthors (2020) [28] presented the Covid-19 Optimizer Algorithm (CVA) to simulate the virus distribution process in several countries around the globe and make decisions by governments and authorities in practicing lockdowns. Al-Betar and coauthors (2021) [29] developed the CHIO algorithm and evaluated it using well-known benchmark continuous test functions such as sphere, Schewefel’s, step, noise functions, etc. And Alweshah and coauthors (2022) [30] incorporated a greedy crossover operator in CHIO for feature selection in medical diagnosis. Fuchigami and Prata (2023) [31] adapted the Covid-based algorithms to the flow shop with earliness and tardiness minimization and anticipation of the common due date with encouraging results.

Particle Swarm Optimization (PSO) is one of the latest evolutionary optimization algorithms inspired by nature. It simulates the behavior of particles to find an optimal solution. Each particle has two attributes – position and velocity, keeping a memory to track its best position. Also, the global best position by the whole swarm is kept so far. PSO differs from other evolutionary methods as it avoids the use of filtering operations such as crossover and/or mutation, and all individuals are maintained through the search procedure so that information is socially shared to direct the movements toward the best position in the search space. Among the advantages of PSO are its simple structure, accessibility for practical applications, ease of implementation, speed to get the solutions, and robustness [32-33]. A review of swarm algorithms applied to discrete optimization problems was published by Krause and coauthors (2013) [34]. Recent applications of PSO in scheduling problems can be found in [32, 35-37].

The Fireworks Algorithm (FWA) is an intelligent metaheuristic proposed by Tan and Zhu (2010) [38] to conduct the global search by mimicking the phenomenon of a fireworks explosion. The algorithm is featured by an explosive search manner and a framework enabling multiple populations to interact. Since it was proposed, the FWA has attracted considerable research interest, widely applied in real-world optimization problems, and shown to be superior or competitive in several fields [39].

Several applications, variants, improvements, and analyses can be found in the book of Tan (2015) [40] and improvements for global optimization were presented by Li and coauthors (2017) [41]. A discrete multi-objective FWA for flow shop scheduling with sequence-dependent setup times was proposed by He and coauthors (2019) [42] and an improved version was implemented for the hybrid flow shop problem by Pang and coauthors (2020) [43].

The proper tuning of the algorithm-specific parameters is a very crucial factor that affects the performance of metaheuristics and improper tuning can either increase the computational effort or yield local optimal solutions. Considering this fact, Rao and coauthors (2011) [44] introduced the Teaching-Learning Based Optimization (TLBO) algorithm which does not require any algorithm-specific parameters. It is a stochastic algorithm inspired by the teaching-learning process in a classroom, where learners are regarded as search solutions. It is performed in two stages: in the teaching phase, learners improve their knowledge levels from the difference between the teacher and the mean of the class; and in the learning phase, learners interact with the group to keep learning.

A survey of the TLBO algorithm and applications was published by Zou and coauthors (2019) [45]. TLBO algorithm has gained wide acceptance among optimization researchers and was applied to flow shop and job shop scheduling problems by [46-48] outperforming other metaheuristics.

Keeping in view the rationale of the TLBO, the same author Rao (2016) [49] proposed Jaya, another algorithm-specific parameter-less metaheuristic even simpler to apply. Having only one phase, 'Jaya' is a Sanskrit word meaning victory and strives to become 'victorious' by reaching the best solution.

According to He and coauthors (2021) [50], there are many advantages of the Jaya algorithm. Mainly, it employs only one search operator and does not have specific parameters, dismissing the calibration effort. Parameter-less algorithms usually have more stable performance. The whole procedure of Jaya is easy to implement computationally, reducing the computational complexity and allowing good solutions in fewer evaluations. The rationale behind the Jaya algorithm is that solutions must move towards the best solutions and escape the bad ones, improving the convergence mechanism. Variants of the algorithm and its engineering applications can be found in the book of Rao (2019) [51] and seven real-world engineering optimization problems were solved by an enhanced version of Jaya algorithm by Zhang and coauthors (2021) [52].

Given the mentioned advantages, there has been growing interest in applying Jaya to deal with complex optimization problems. Specifically in scheduling, Buddala and Mahapatra (2018) [53] applied TLBO and Jaya algorithms to the flexible flow shop problem with makespan minimization. Mishra and Shrivastava (2020) [54] showed that Jaya algorithm outperforms other metaheuristics, such as hybrid genetic algorithm, hybrid differential evolution, hybrid particle swarm optimization, hybrid backtracking search algorithm and TLBO, in the permutation flow shop problem with makespan minimization. And Fan and coauthors (2021) [55] proposed a hybrid Jaya algorithm with tabu search for solving the flexible job shop problem with makespan minimization.

In terms of multi-objective problems, Mishra and coauthors (2019) [56] applied Jaya algorithm in a permutation flow shop problem with minimization of makespan and tardiness cost under due date constraint. He and coauthors (2021) [50] adapted it for scheduling the job shop problem with multi-objective of makespan, flow time, and tardiness. And Caldeira and Gnanavelbabu (2021) [57] proposed a discrete Jaya algorithm for makespan, total workload machines, and workload of critical machines in a flexible job shop problem.

PROBLEM DESCRIPTION

In the considered problem, let a set of n independent jobs to be processed on m machines in series. Each job j, j=1,…,n, is available for processing at time zero and requires p_{jk} time units, j=1,…,n, k=1,…,m, for processing on each machine k, k=1,…,m, which is known beforehand. It is also assumed that: all jobs are available for processing; all jobs follow the same predefined order of operations; no preemption or interruption is allowed; no job can be processed by more than one machine at the same time, and no machine can process more than one operation at the same time; all jobs must visit all machines with strictly positive processing time on all the machines (no skipping machines). Let $\pi=(J_{[1]}, J_{[2]}, \ldots, J_{[n]})$ be the sequence of n jobs. The subscript $[h]$ denotes the job occupying in the h^{th} position in the sequence π. For a given sequence π, the completion time of the job at the position h on machine k is denoted by C_{hk} and on the last machine (m^{th}) is C_{mk}. The objective is to look for the sequence π that minimizes the cost function Q. The optimal solution consists of a sequence π* with the optimal cost Q^*.

Among the different mathematical formulation strategies for flow shop problems, we proposed a mathematical model based on position variables, which is known as the most appropriate for many scheduling problems (cf. e.g., [58-62]). This formulation dispenses the parameter *bigM* that influences the computational efficiency of the model execution by the optimization solver. Ren and coauthors (2016) [1] presented a similar model but with no computation experimentation results about it. This formulation with position-based variables is also derived from [63] for the classic flow shop.

Let the decision variable $x_{jh} \in \{0,1\}$, j=1,…,n, h=1,…,n, equal to 1 if job j is scheduled at the position h of the sequence and 0 otherwise. The mathematical model is given as follows.

$$\text{Min} \quad Q = \sum_{h=1}^{n} C_{hm}^{2} \tag{1}$$

$$\text{subject to} \quad \sum_{h=1}^{n} x_{jh} = 1, \qquad j=1,\dots,n, \tag{2}$$

$$\sum_{j=1}^{n} x_{jh} = 1, \qquad h=1,\dots,n, \tag{3}$$

$$C_{11} = \sum_{j=1}^{n} p_{j1} x_{j1}, \tag{4}$$

$$C_{hk} \geq C_{h,k-1} + \sum_{j=1}^{n} p_{jk} x_{jh}, \qquad h=1,\dots,n,\ k=2,\dots,m, \tag{5}$$

$$C_{hk} \geq C_{h-1,k} + \sum_{j=1}^{n} p_{jk} x_{jh}, \qquad h=2,\dots,n,\ k=1,\dots,m, \tag{6}$$

$$x_{jh} \in \{0,1\}, C_{hk} \geq 0, \qquad j=1,\dots,n,\ h=1,\dots,n,\ k=1,\dots,m. \tag{7}$$

Equation (1) represents the objective function. Constraints (2) and (3) are the assignment problem constraints, in which the set (2) guarantees that each job is assigned to just one position in the sequence and the set (3) ensures that each sequence position contains just one job. Constraints (4) to (6) impose the rules for the completion time of each job on each machine. Constraint (4) simply indicates that the completion time of the first job on the first machine must be at least its processing time. Constraints (5) express that between the completion time of a job on two consecutive machines, there must be enough time for the job to be processed on the first one. Constraints (6) indicate that between the completion times of consecutive jobs on a machine, there must be enough time for the first one to be processed. Finally, constraints (7) are the domain of the variables. Regarding this model size complexity, its number of variables is $n(n+m)$, in which n^2 are binary and nm are continuous, and $2n(m+1)+1$ constraints.

PROPOSED SOLUTION METHODS

We proposed Covid and CHIO algorithm based on Fuchigami and Prata (2023) [31], by adapting the calculation of the total quadratic completion times of jobs and the mutation process. Also, the parameters of the metaheuristics presented were calibrated specially for the present scheduling problem.

Covid algorithm

The Covid algorithm employs the concepts of the real pandemic with four statuses of the population individuals: susceptible, infected, recovered, and dead. The infection process is given by the mutation applied in individuals of different statuses depending on predetermined probabilities. The mutation operator consists of swapping two jobs chosen randomly and repeating this process an equally random number of times from 1 to 5. This mechanism balances the intensification and diversification of the metaheuristic. For the proposed Covid algorithm, consider the notation described in Table 1.

Table 1. Covid algorithm's notation

Notation	Description
termination	Termination criterion set as the CPU time
popsize	Size of the population
p_die	Fatality rate
p_reinfection	Probability of a recovered individual being reinfected
p_isolation	Social isolation rate of the population; it helps to reduce the exponential growth of the infected population
status[i]	Status of each individual i, $i = 1, \dots, popsize$ (0: susceptible, 1: infected, 2: recovered, 3: dead)

The pseudocode of Covid algorithm is presented in Algorithm 1.

CHIO algorithm

In the CHIO algorithm, herd immunity refers to a situation where enough people in a population have immunity to the infection to be able to effectively stop the disease from spreading. It does not depend on whether the immunity comes from vaccination or from the people who had the disease. So, the population individuals are classified into three types: susceptible, infected, and immune (or recovered) individuals.

Algorithm 1 Covid algorithm

```
Input: $p$
Output: $\pi, Q$
procedure covid
Initialize popsize, p_die, p_reinfection, p_isolation, termination
Generates the initial population randomly with $status$ = 0, 1 or 2 // susceptible, infected or recovered
for $i = 1 : popsize$ do
    if $status[i] = 1$ and rand() $< p\_die$ then
        $status[i] = 3$
    end if
end for
while time() $< termination$ do
    for $i = 1 : popsize$ do
        if $status[i] = 0$ then
            if rand() $< p\_isolation$ then
                $status[i] = 4$
            end if
            apply mutation in the individual $i$
        else if $status[i] = 1$ then
            if rand() $< p\_die$ then
                $status[i] = 3$
            end if
            apply mutation in the individual $i$
        else if $status[i] = 2$ then
            if rand() $< p\_reinfection$ then
                $status[i] = 1$
            end if
            apply mutation in the best individual of the population
        else
            generates a new individual randomly with $status[i] = 0$
        end if
        if new individual is better than individual $i$ then
            replace individual $i$ by the new individual
        end if
        if fitness of $i >$ mean fitness of the population then
            $status[i] = 2$
        end if
    end for
end while
return best individual and fitness
end procedure
```

A herd immunity population (HIP) is randomly generated with C_0 initial infected cases and other susceptible individuals. Then according to a basic reproduction rate (BRr), the pandemic evolves to infected, susceptible or immune new cases. Also, an age vector (A_i) is considered to limit the generation number of each individual in the population. For CHIO, we define the notation of Table 2.

Table 2. CHIO algorithm's notation

Notation	Description
termination	Termination criterion set as the CPU time
C_0	Number of initial infected individuals
BRr	Basic reproduction rate
max_age	Maximum age of infected individuals
popsize	Population size
HIP[i,j]	Herd immunity population, $i = 1, \dots, popsize$, $j = 1, \dots, n$
S[i]	Status of each individual i, $i = 1, \dots, popsize$ (0: susceptible, 1: infected, 2: immuned)
A[i]	Age vector of each individual i, $i = 1, \dots, popsize$
F[i]	Fitness (objective function) of each individual i, $i = 1, \dots, popsize$
∆F	Mean value of the population objective function

For the perturbation scheme to generate new solutions, we used here the pair-swap operation to keep the comparability of the algorithms. Thus, we define the number of swaps performed at each mutation operation as a random selection in the interval [1, 5]. The complete algorithm of the CHIO is introduced in Algorithm 2.

Algorithm 2 CHIO algorithm

```
Input: $p$
Output: $\pi, Q$
procedure chio
Initialize $C_0$, popsize, BRr, max_age, termination
Set $S[i] = 0, A[i] = 0, i = 1, ..., popsize$
for $i = 1 : popsize$ do
    generate a random permutation $HIP[i, j], j = 1, ..., n$ for each individual $i$ and calculate $F[i]$
end for
Generate randomly the $C_0$ infected individuals
while time() $< termination$ do
    for $i = 1 : popsize$ do
        $r$ = rand() // $U[0, 1]$
        if $r < BRr/3$ then
            $is\_corona = true$
            select randomly the transmitter individual for mutation (individual $indiv$)
            if no transmitter individual then
                consider the current individual $i$ for mutation
            end if
        else if $r < 2BRr/3$ then
            $is\_corona = false$
            select randomly the susceptible individual for mutation (individual $indiv$
            if  no susceptible individual then
                consider the current individual $i$ for mutation
            end if
        else if $r < BRr$ then
            $is\_corona = false$
            find the best immune individual for mutation (individual $indiv$
        else
            consider the current individual $i$ for mutation (individual $indiv$
            if $S[indiv] = 1$ then
                $is\_corona = true$
            else
                $is\_corona = false$
            end if
        end if
        generate new individual applying mutation in the individual $indiv$
        calculate the fitness $F_{new}$ of the new individual $indiv$
        if $F_{new} \leq F[i]$ then
            update the current individual $i$ with the new $indiv$
        end if
        if $F_{new} < \Delta F$ and $S[indiv] = 0$ and $is\_corona$ then
            $S[indiv] = 1; A[indiv] = 1$
        else if $F_{new} > \Delta F$ and $S[indiv] = 1$ then
            $S[indiv] = 2; A[indiv] = 0$
        end if
        if $A[indiv] \geq max\_age$ and $S[indiv] = 1$ then
            generate new individual applying mutation in the individual $indiv$
            $S[indiv] = 0; A[indiv] = 0$
            update the fitness $F[indiv]$
        end if
    end for
end while
return best individual and fitness
end procedure
```

RESULTS AND DISCUSSION

This section describes the experimental setup and presents the analysis of the results, as well as comparisons with previous algorithms to assess the performance of the proposed methods. Also, it was presented the performance profiles of the compared metaheuristics.

Experimental setup

To conduct a fair scientific analysis, the same computational resources and conditions were observed throughout the experiments. The experiments were carried out using a Pentium Intel Core i7 with ~2.0GHz processor, 16.0GB RAM and Windows Operation System. Codes were implemented in the syntax of the Julia programming language (a high-performance programming language for modern technical computing) version 15.5.3-1 using the Atom editor with the JuMP modeling language (Julia for Mathematical Programming) embedded for mathematical optimization. The branch-and-cut algorithm included in the Gurobi optimization solver version 9.1.0 was used for the mathematical model with all default parameters and CPU time limited to 2 hours (7200 seconds).

All methods were implemented and executed using five benchmark datasets including two industrial settings, making up 620 problem instances solved, referred to:

- **Ren:** the industrial data set of Ren and coauthors (2016) [1], of a hot-rolling process in steelmaking composing a two-machine flow shop wherein hot and cold slabs are first reheated to the rolling temperature by the heat furnace. The slabs are then processed by the rolling machine. Ten data sets with $n \in \{40, 60, 80, 100\}$ and job weights in U[1, 10] were considered.
- **Bai:** the industrial data set of Bai and coauthors (2017) [17], of a gearbox fabrication, an important mechanical component in wind turbine generator. It consists of a flow shop with eight steps in series, such as parts cleaning, hot assembling, cooling, debugging, testing, lubricant injecting, painting, and final assembling. Ten data sets with $n \in \{50, 100, 150, 200\}$ were employed.
- **Taillard:** the known testbed of Taillard (1993) [64], containing 120 instances divided into 12 groups with different sizes n x $m \in$ {20x5, 20x10, 20x20, 50x5, 50x10, 50x20, 100x5, 100x10, 100x20, 200x10, 200x20, 500x20}.
- **Small VRF:** the small VRF hard instances of Vallada and coauthors (2015) [65], comprising 12 groups with $n \in \{10, 20, 30, 40, 50, 60\}$ and $m \in \{5, 10, 15, 20\}$, with the full orthogonal combinations of n x m.
- **Large VRF:** the large VRF hard instances of Vallada and coauthors (2015) [65], also composed by 12 groups with $n \in \{100, 200, 300, 400, 500, 600, 700, 800\}$ and $m \in \{20, 40, 60\}$, with the full orthogonal combinations of n x m.

In addition to the mathematical model formulated, six metaheuristics were implemented to compare the efficiency of the proposed algorithms, especially for the large-size instances: FWA based on Tan and Zhu (2010) [38], PSO of Tasgetiren and coauthors (2007) [33], Jaya of Mishra and Shrivastava (2020) [53], TLBO of Baykasoglu and coauthors (2024) [47], IGA of Pan and Ruiz (2012) [2], and DE of Ren and coauthors (2016) [1].

For a fair comparison, as in [2, 66], the same maximum elapsed CPU time limit of t=30nm milliseconds was adopted in all metaheuristics as a termination criterion. The choice of this stopping criterion is motivated by the fact that all algorithms are coded in the same programming language, share most library functions and data structures, and are executed on the same computer. For each instance, five independent runs were conducted for each algorithm as considered by Pan and Ruiz (2012) [2].

Both proposed and reference metaheuristics were calibrated to the problem approached through the tuning method of Montgomery (2017) [67]. All possible parameter configurations of each algorithm were tested independently with a termination criterion set to a maximum elapsed CPU time t = 10nm milliseconds. Each algorithm was tested with a set of 10 randomly generated instances different from each of the benchmark sets.

It is of paramount importance to separate the calibration benchmark from the final testing benchmark. Calibrating algorithms with the same instances that will be used later for computational results and comparisons constitutes poor practice. By separating the calibration and implementation instances we avoid over-calibration and overly optimistic results, where those excellent performances might not be replicable to real instances or other benchmarks Pan and Ruiz (2012) [2]. The number of jobs and machines for each calibration instance was $n \in \{5, 10, 25, 50\}$ and $m \in \{3, 5, 10\}$ and the processing times for each instance were obtained from a discrete uniform distribution in the interval [1, 99]. After the calibration process, the recommended values for each parameter are: Covid: *p_die* = 0.05, *p_reinfection* = 0.01, *p_isolation* = 0.6, *popsize* = 50; CHIO: C_0 = 5, *popsize* = 30, *BRr* = 0.05, *max_age* = 300; FWA: *n_firework* = 3, *m_spark* = 80, *n_spark_gaussian* = 10, a = 0.06, b = 0.6, A_{max} = 30; PSO: *popsize* = 100, w_{max} = 1, w_{min} = 0.45, c_1 = 2, c_2 = 1; Jaya: *popsize* = 200; TLBO: *popsize* = 30; IGA: λ = 0.4, d = 8.

Parameters were chosen based on the lowest average relative percentage deviation, calculated as RPD = 100(Q-Q^*)/Q^*, the result for each candidate value. In the equation, Q is the objective function value of an instance by a given algorithm, and *Q** is the best objective function value by any of the solution methods (algorithms and/or model) for the same instance.

Analysis of the results

The statistical comparison for the industrial benchmarks, Ren and Bai, for the average relative percentage deviation (ARPD) in relation to the optimal solution from the mathematical model is presented in Figures 1 and 2. Covid and CHIO presented ARPD zero for the Bai benchmark in all options of instance sizes, i.e., they provided the optimal solution in all the cases. For Ren benchmark, these two algorithms achieved ARPD of only 0.01%. The third best was IGA in the Ren data set, with RPD of 0.38%, and FWA the in Bai data set, with 0.06%. PSO was the worst in all the cases.

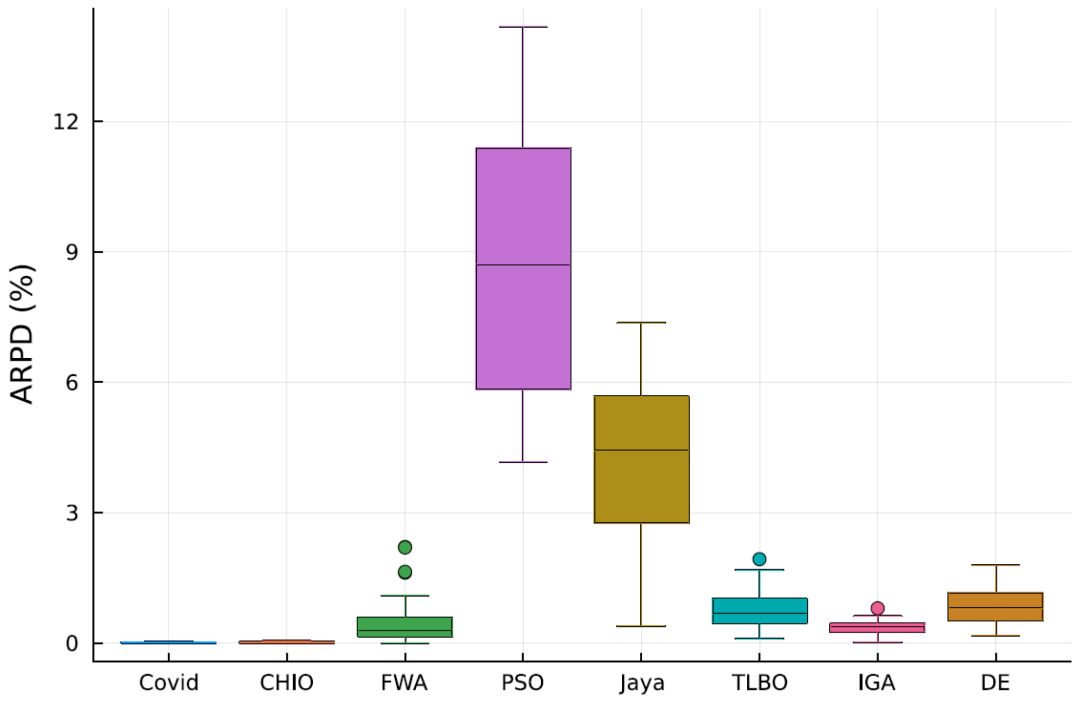

Figure 1. Statistical comparison among the metaheuristics for the industrial Ren benchmark.

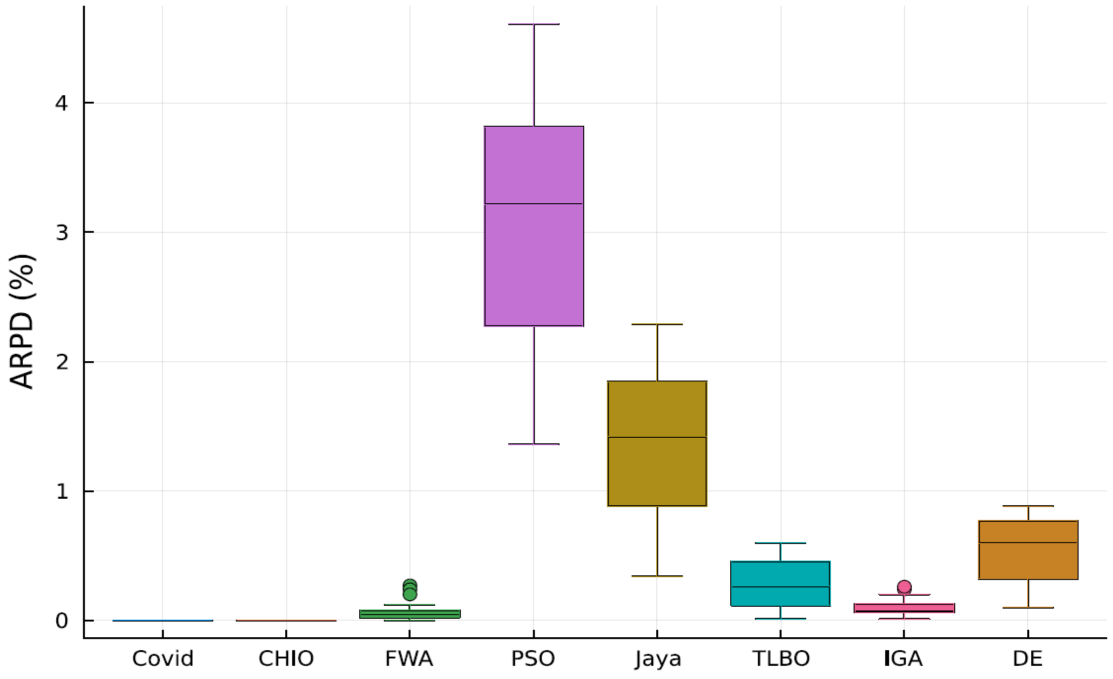

Figure 2. Statistical comparison among the metaheuristics for the industrial Bai benchmark.

For the benchmarks of the literature, Figure 3 illustrates the statistical comparison of each metaheuristic for ARPD in relation to the best-found solution for the Taillard benchmark, again confirming the superiority of CHIO and Covid's results. The mathematical optimization solver could achieve the optimal solution only for the Taillard instances with 20 jobs and 5 machines. The metaheuristic CHIO achieved the best results in instances up to 100 jobs and Covid algorithms outperformed the others in the largest instances, with 200 and 500 jobs.

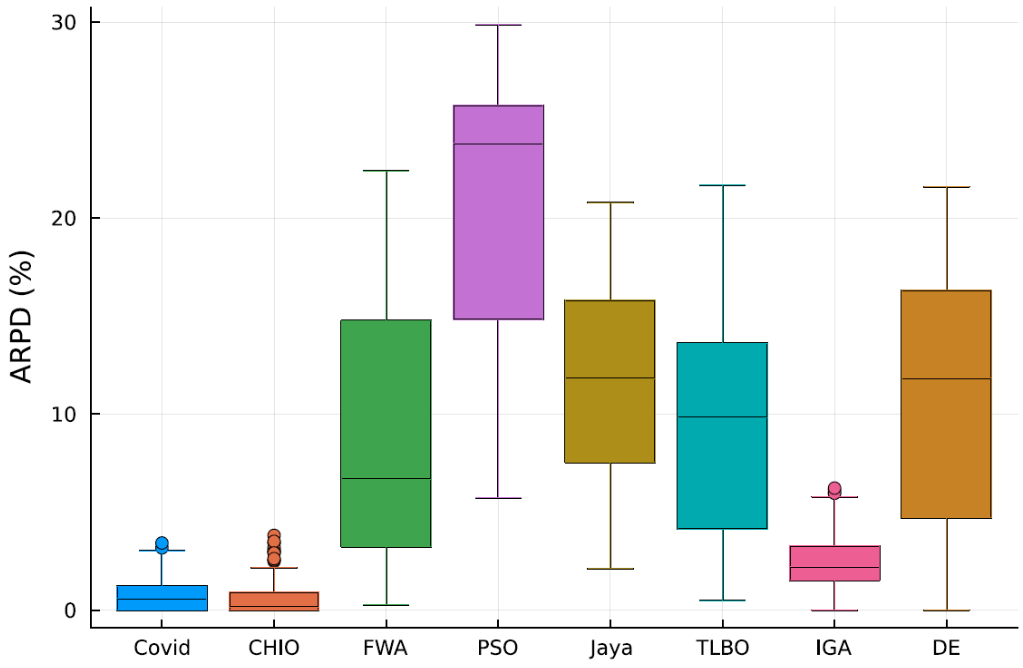

Figure 3. Statistical comparison among the metaheuristics for the Taillard benchmark

Figures 4 and 5 presents the results of the metaheuristics for the VRF benchmarks related to the best-found solution. In the case of Small VRF, the Model achieved optimality up to instances with 20 jobs and 5 machines. Covid, CHIO, TLBO, IGA, and DE provided the optimal solution for instances up to 10 jobs and 20 machines and FWA yielded the optimal solution for 10 jobs and 15 and 20 machines. The best results in all the instances of Small VRF were given by CHIO, with an average RPD of only 0.23%.

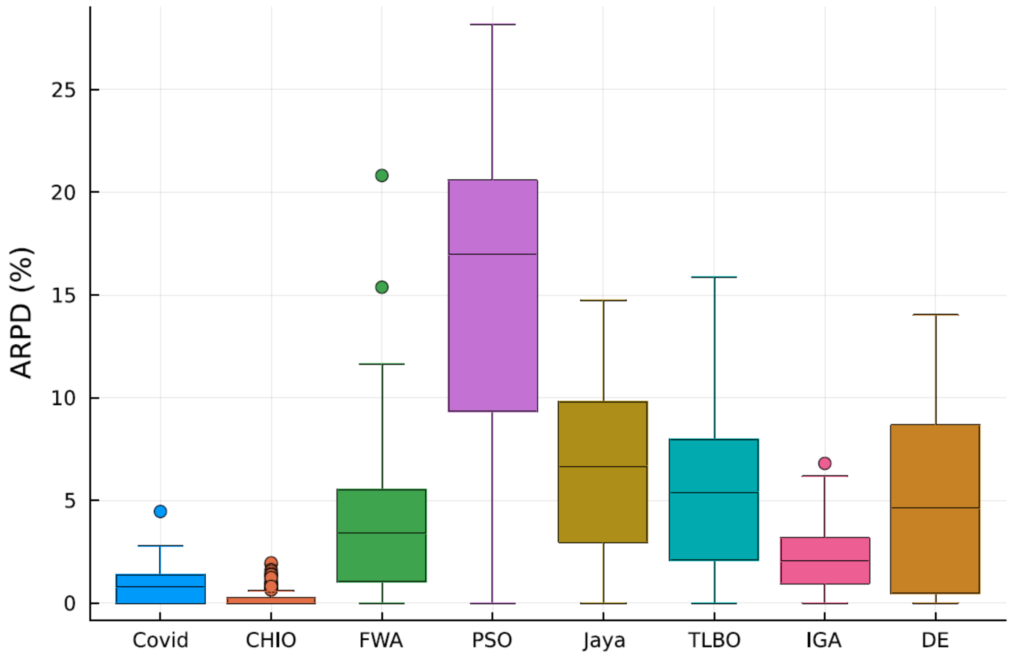

Figure 4. Statistical comparison among the metaheuristics for the Small VRF benchmark.

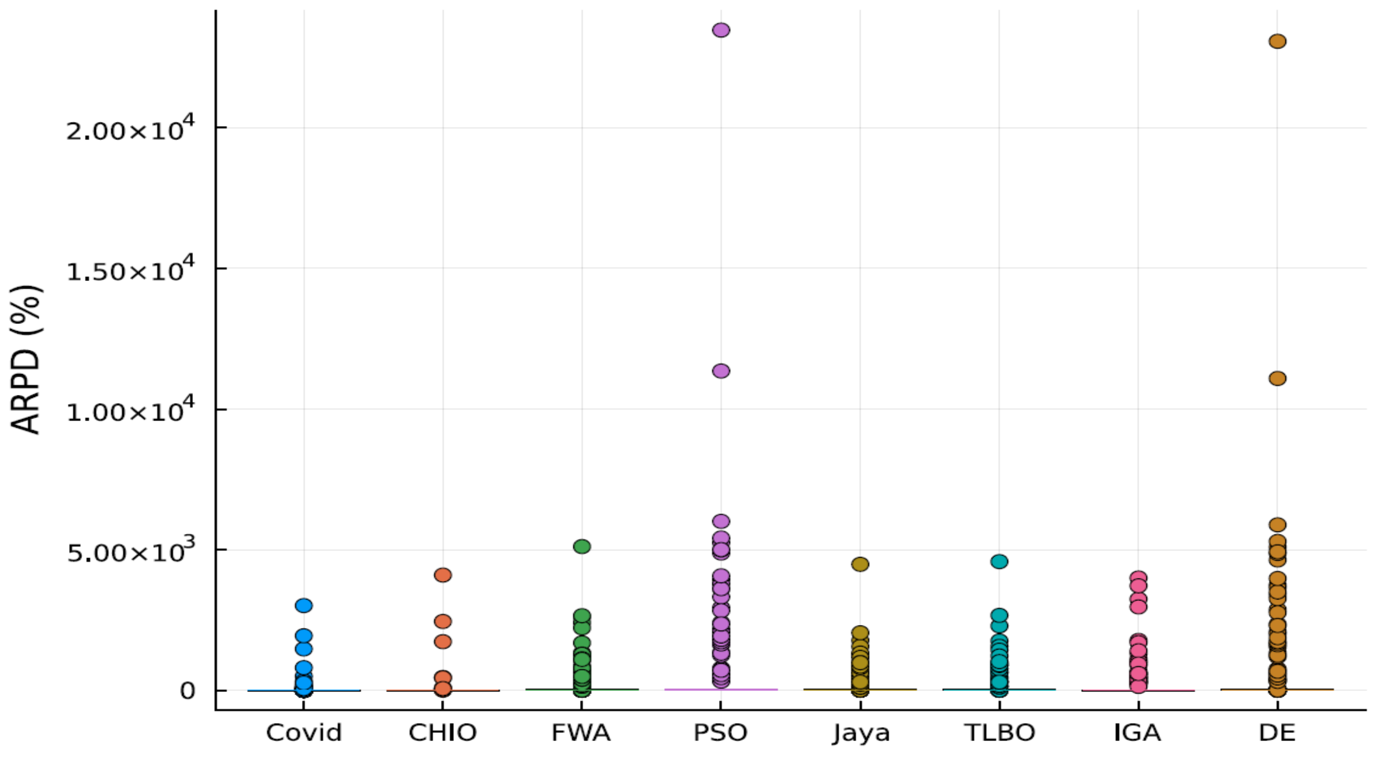

Figure 5. Statistical comparison among the metaheuristics for the Large VRF benchmark.

For the Large VRF benchmark, Covid algorithm outperformed the others in 18 of the 24 options of instance sizes, with zero RPD in 16 of them. CHIO was the best in the other 6 of the 24 instance sizes. In all instances of the Large VRF, the variation of the RPD is considerable, especially in the worst metaheuristics: PSO, DE, and FWA.

Performance profiles

Finally, we provide a visual representation of each method's relative performance through performance profiles as proposed by Doland and Moré (2002) [68]. This evaluation method presents a clear indication of the relative performance of each solver as a means of providing objective information when comparing optimization algorithms [69]. Some recent and interesting works on various scheduling problems that successfully took advantage of this method in its evaluation process are [58, 70, 71].

The performance profile of a method is the cumulative distribution function, which indicates the probability of a performance ratio is within the interest factor (*f*). Therefore, the *P*(*f*) indicates the percentage of instances in which the method achieves in *f* times the best value of the measure. Figures 6 to 10 present the performance profiles of all solution methods for each benchmark set and Table 3 summarizes information of each method's performance profile extreme values.

Figures 6 and 7 present the performance profiles for the industrial benchmarks of Ren and Bai. For Ren set of instances, it is possible to identify three groups of methods which have similar curve behavior. The first group includes the Model, Covid, and CHIO methods which performed the best achieving at least 12.5% of overall best solutions and its solutions are, at most, 0.5% worse than the best one. The second group consists of FWA, TLBO, IGA, and DE which did not achieve any overall best solution, *P*(1)=0, and its worst solutions are less than 2.21% worse than the best one. The third group contains just PSO and Jaya which had the worst performance with *P*(1)=0 and worst solutions of, respectively, 14.18% and 7.36% worse than the best one.

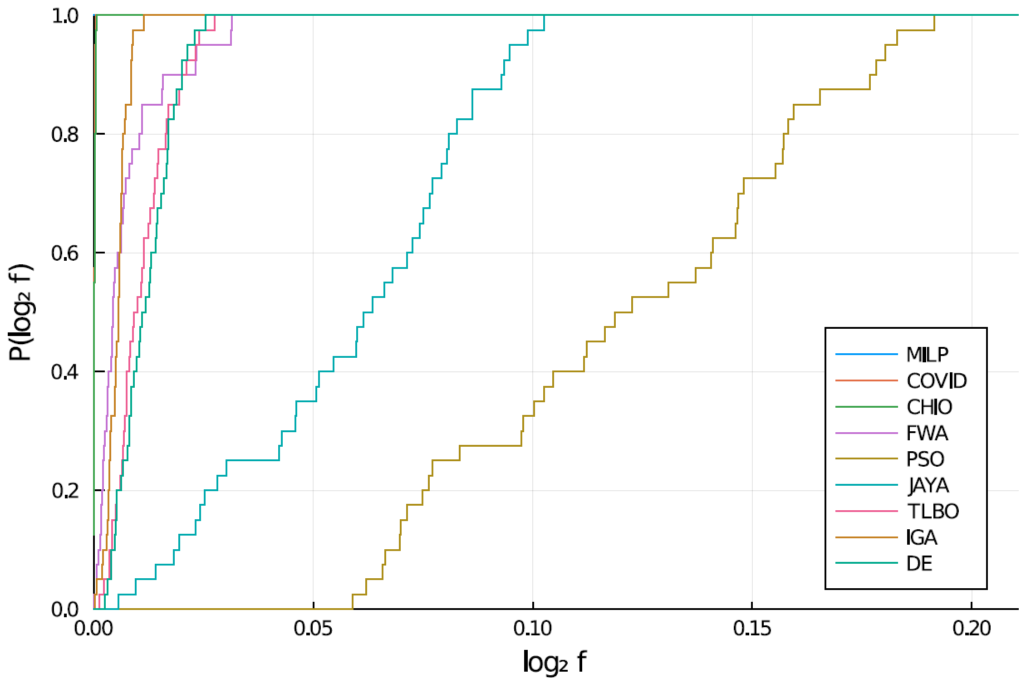

Figure 6. Performance profiles of algorithms and model for the industrial Ren benchmark.

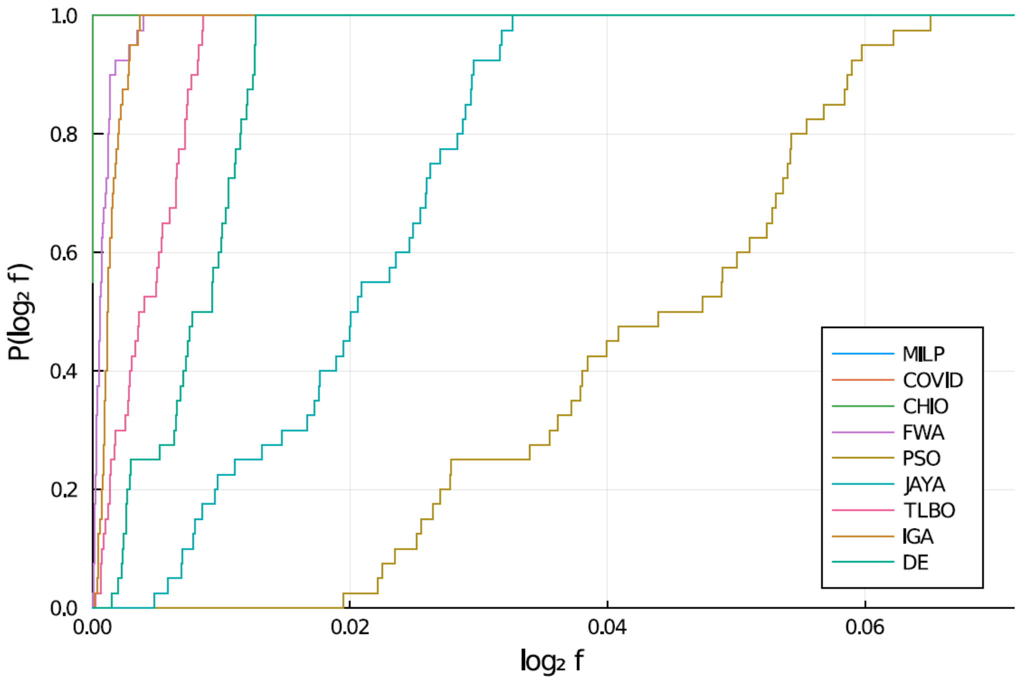

Figure 7. Performance profiles of algorithms and model for the industrial Bai benchmark.

Since the Model and Covid achieved optimal solutions for all instances of Bai benchmark, its series in Figure 7 is a straight line in $P(\log_2 f)=1$. CHIO found optimal solutions for 55% of instances and its worst solution is only 0.001% worse than the best one. The remainder of the algorithms have similar performance compared with Ren benchmarks. FWA, TLBO, IGA, and DE did not achieve optimal results and their worst solutions are, at most, 0.88% worse than the best one. PSO and Jaya also found no optimal solutions and their worst performance are 4.61% and 2.29% worse than the best one, respectively.

For both Ren and Bai sets, note that FWA, PSO, Jaya, TLBO, IGA, DE methods' best solutions are only with $f>1$, i.e., only the Model, Covid and CHIO achieved some overall best solution, $P(1)\geq 0.125$ (see Table 3 for more detail).

Figure 8 presents the performance profile of each method for Taillard benchmark set. Covid and CHIO methods outperform all remainder methods with respect to both percentages of best results (27.50% and 43.33%, respectively) and how worse is its worst solutions (3.43% and 3.82%, respectively).

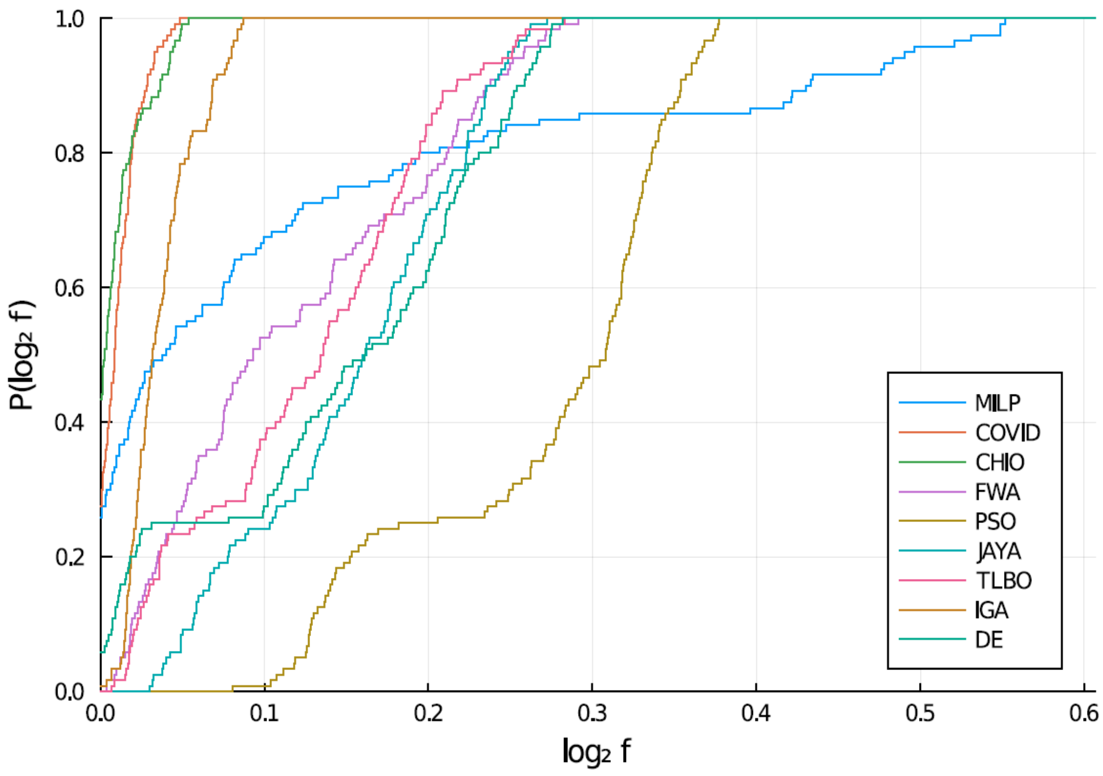

Figure 8. Performance profiles of algorithms and model for the Taillard benchmark.

Although the Model achieved overall best results in 25.83% of instances and 10.83% of optimal solutions, its worst result is 46.57% worse than the best one (outperformed by all other methods) and this is represented in Figure 9 as the Model's curve behavior as it starts higher than other algorithms and follows as a concave function until near $P(\log_2 f)$=0.4. IGA and DE are the last two that achieve some overall best solution (0.83% and 5.83%, respectively) and 6.23% and 21.59%, respectively, regarding the worst solution. The remainder methods did not achieve some overall solutions and had inferior performance regarding the worst solutions.

Figures 9 and 10 present the performance profile for VRF benchmarks. For Small VRF, Figure 9 shows all 9 methods achieved the overall best solution in at least 9.58% of instances. In this set, although the model found optimal values for 20.83% of instances CHIO achieved overall best solutions in 64.58% of instances. Also, Covid and CHIO's worst solutions (4.46% and 1.96% worse than the best ones, respectively) are better than the Models' worst solutions (6.48% worse than the best ones, respectively). It is possible to notice that the remainder algorithms increase its percentage of solutions within an interest factor in a linear-like way, except for PSO method its worst solution is 28.19% worse than the best one.

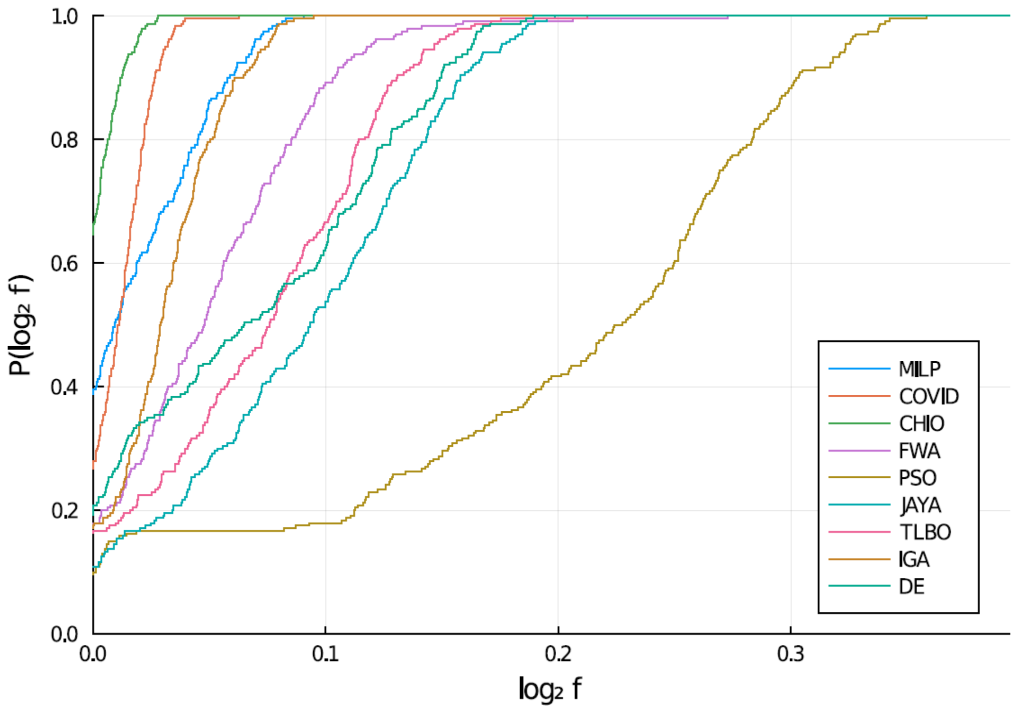

Figure 9. Performance profiles of algorithms and model for the Small VRF benchmark.

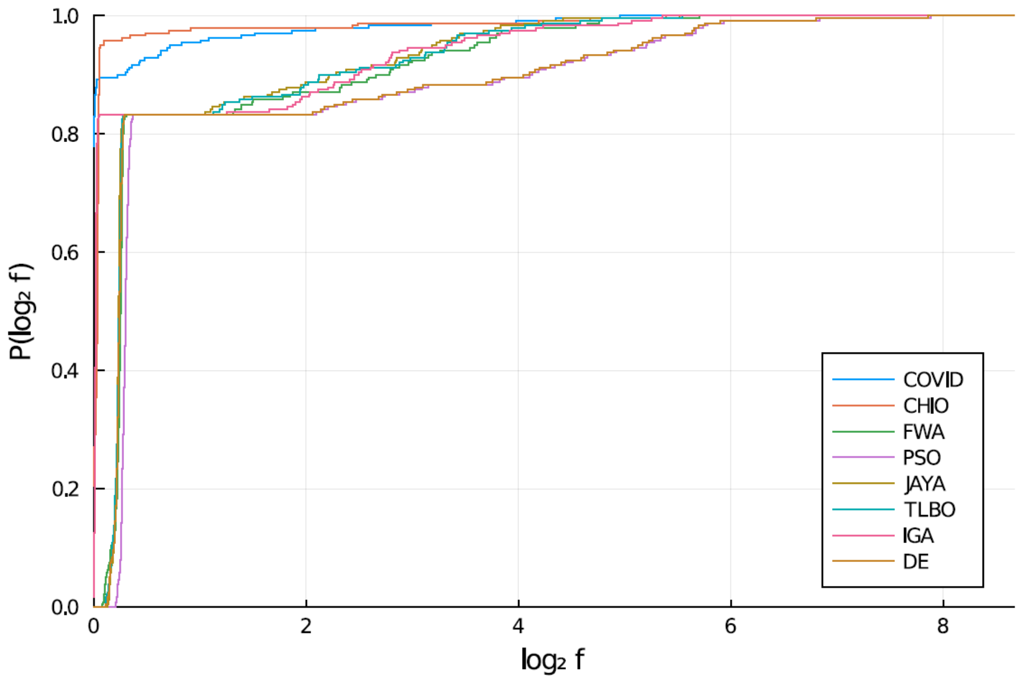

Figure 10. Performance profiles of algorithms and model for the Large VRF benchmark.

Figure 10 illustrates for the Large VRF benchmark only the metaheuristics' performance profiles since the Model was not executed with this set. Notice that all methods accumulate more than 80% of its solutions within *f*≤2 while the lowest worst solution, among all methods, is *f*=31 times worse than the best one. This comparison shows the fact that, for about 80% of instances, all methods had similar satisfactory performance but, for the remainder 20% of instances, its performance got much worse, even achieving values of about 235 times worse than the best solution.

Table 3 summarizes the extreme values of each method's performance profile among all five benchmark sets. Unlike the figures, in the table we do not use log values. The *P*(1) columns stand for the value of the probability function *P*(*f*) when *f*=1, i.e., the percentage of times that the method achieved the best value among all benchmark instances. Since some methods did not achieve any best solution among all instances, we represented this scenario as '-'. The f columns stand for the value of the interest factor when *P*(*f*)=1, i.e., the method is, at most, (*f*-1)%, or *f* times, worse among all methods. We highlight in bold the method that achieved the best value for each statistic in the benchmark. Note that there are no Model's values of *P*(1) and *f* for Large VRF benchmark since it was not implemented for these instances set.

Table 3. Performance profiles extreme values of *P*(1) and *f* for each method and benchmark set

	Ren		Bai		Taillard		Small VRF		Large VRF	
	***P*(1)**	***f***	***P*(1)**	***f***	***P*(1)**	***f***	***P*(1)**	***f***	***P*(1)**	***f***
Model	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.2583	1.4657	0.3875	1.0648	-	-
Covid	0.1250	1.0003	**1.0000**	**1.0000**	0.2750	**1.0343**	0.2667	1.0446	**0.7792**	**31.1372**
CHIO	0.1250	1.0005	0.5500	1.0000	**0.4333**	1.0382	**0.6458**	**1.0196**	0.2042	41.9907
FWA	-	1.0221	-	1.0027	-	1.2240	0.1750	1.2082	-	52.1172
PSO	-	1.1418	-	1.0461	-	1.2987	0.0958	1.2819	-	235.7360
Jaya	-	1.0736	-	1.0229	-	1.2079	0.1083	1.1475	-	45.8130
TLBO	-	1.0193	-	1.0060	-	1.2168	0.1625	1.1588	-	46.7723
IGA	-	1.0080	-	1.0026	0.0083	1.0623	0.1708	1.0680	0.0167	40.9434
DE	-	1.0179	-	1.0088	0.0583	1.2159	0.1917	1.1402	-	231.7476

Notice the Model, Covid and CHIO's performance profile extreme values for Ren benchmark. Since the Model achieved optimal solutions for all instances, its percentage of best solutions among all 40 instances is 100% and accumulated all solutions within the interest factor of *f*=1. Regarding the CHIO method, the best solution was achieved in 12.5% of the 40 instances and CHIO's worst solution is 0.0005%, or 1.0005 times, worse than the best solution. Finally, Covid also achieved the best solution in 12.5% of instances and its worst solution is 0.0003%, or 1.0003 times, worse than the best solution.

In general, it is observed that both Covid and CHIO achieved overall best solutions (optimal solutions in some cases) for all benchmark sets. While considering the Model results for Ren and Bai sets, in which it achieved optimal solutions for all instances, Covid and CHIO sum 5 and 3 best extreme values, respectively. When not considering the Model results, Covid and CHIO show the best extreme values of 7 and 4, respectively. Except for the Small VRF set, FWA, PSO, Jaya, and TLBO failed to achieve an overall best solution. While IGA and DE found the overall best solutions for Taillard and Small VRF sets, IGA found it for Large VRF benchmark.

CONCLUSION

In this work, we solved the flow shop problem with total quadratic completion time minimization, which is a trade-off between makespan and flowtime optimization. We implemented nine solution methods, among metaheuristics and a mathematical model, in five benchmark sets of instances with varied sizes (two industrial and three from the literature). All results were statistically analyzed through relative percentage deviation and the prominent performance profiles.

Our Covid and CHIO algorithms outperformed all other metaheuristics tested, including Ren and coauthors (2016) [1]'s DE and the state-of-art IGA, of Pan and Ruiz (2012) [2], in all five benchmarks. CHIO and Covid algorithms had only 0.23% and 0.41% of average relative deviation from the optimal solution, respectively. The mathematical model proved optimality for all instances of the industrial data sets and for size 20x5 of Taillard and up to 20x5 of small VRF. So, this is the optimality threshold for the problem size that Gurobi Optimizer can solve the treated problem. PSO metaheuristic provided the worst result and Jaya the second worst in all the cases. FWA, PSO, Jaya, TLBO methods failed in achieving some overall best solution in four of the five benchmarks.

The main contributions of this work are: (i) investigation of a scheduling problem little addressed in the literature with considerable practical importance; (ii) state-of-the-art update of publications related to total quadratic completion time; (iii) proposition of a metaheuristic better than the previously employed to solve the problem; (iv) enabling the resolution of large-size problems in a viable computing time; and (v) performance profiles and statistical analysis of nine solution methods in five benchmark data sets.

Suggestions for future works include the study of quadratic measures for green scheduling problems, extensions such as parallel flow shops and/or hybrid flow shops, warm-start solutions in the mathematical model, and specialized local search stage in the metaheuristics.

Acknowledgments: The authors are grateful to Sao Paulo Research Foundation (FAPESP) for the financial support (grants numbers 2022/05803-3, 2022/10993-6, and 2023/08678-8).
Conflicts of Interest: The authors declare no conflict of interest.

REFERENCES

1. Ren T, Zhao P, Zhang D, Liu B, Yuan H, Bai D. Permutation flow-shop scheduling problem to optimize a quadratic objective function. Eng Optim. 2016;49(9):1589-603.
2. Pan QK, Ruiz R. Local search methods for the flowshop scheduling problem with flowtime minimization. Eur J Oper Res. 2012;222(1):31-43.
3. Fuchigami HY, Rangel S. A survey of case studies in production scheduling: Analysis and perspectives. J Comput Sci. 2018;25:425-36.
4. Bellabai JR, Leela BNM, Kennedy SMR. Testing the performance of Bat-algorithm for permutation flow shop scheduling problems with makespan minimization. Braz Arch Biol Technol. 2022;65.
5. Fuchigami HY, Moccellin JV, Ruiz R. [New priority rules for the flexible flow line scheduling problem with setup times]. Prod. 2015;25:779-90.
6. Prata BA, Fuchigami HY. A genetic iterated greedy algorithm for the blocking flowshop to minimize total earliness and tardiness. J Intell Manuf. 2023.
7. Tyagi N, Varshney RG, Chandramouli AB. Six decades of flowshop scheduling research. Int J Sci Eng Res. 2013;4(9):854-64.
8. Potts CN, Strusevich VA. Fifty years of scheduling: a survey of milestones. J Oper Res Soc. 2009; 60:S41-S68.
9. Gupta JND, Stafford Junior EF. Flowshop scheduling research after five decades. Eur J Oper Res. 2006;169:699-711.
10. Reza-Hejazi S, Saghafian S. Flowshop-scheduling problems with makespan criterion: a review. Int J Prod Res. 2005;43(14):2895-929.
11. Graham RL, Lawler EL, Lenstra JK, Rinnooy Kan AHG. Optimization and approximation in deterministic sequencing and scheduling: a survey. Ann Discrete Math. 1979;5:287-326.
12. Bai D. Asymptotic analysis of online algorithms and improved scheme for the flow shop scheduling problem with release dates. Int J Syst Sci. 2015;46(11):1994-2005.
13. Bai D, Zhang Z. Asymptotic optimality of shortest processing time-based algorithms for flow shop and open shop problems with nonlinear objective functions. Eng Optim. 2014;46(12):1709-28.
14. Cheng TCE, Liu Z. Parallel machine scheduling to minimize the sum of quadratic completion times. IIE Trans. 2004;36(1):11-7.
15. Koulamas C, Kyparisis G. Algorithms with performance guarantees for flow shops with regular objective functions. IIE Trans. 2005;37(12):1107-11.
16. Bai D, Xue H, Wang L, Wu C-C, Lin W-C, Abdulkadir DH. Effective algorithms for single-machine learning-effect scheduling to minimize completion-time-based criteria with release dates. Expert Syst Appl. 2020;156:113445.
17. Bai D, Liang J, Liu B, Tang M, Zhang Z-H. Permutation flow shop scheduling problem to minimize nonlinear objective function with release dates. Comput Ind Eng. 2017;112:336-47.
18. Ren T, Diao Y, Luo X. Optimal results and numerical simulations for flow shop scheduling problems. J Appl Math. 2012;2012:1-9.
19. Xu Z, Sun L, Gong J. Worst-case analysis for flow shop scheduling with a learning effect. Int J Prod Econ. 2008;113:748-53.
20. Wang J-B, Wang M-Z. Worst-case behavior of simple sequencing rules in flow shop scheduling with general position-dependent learning effect. Ann Oper Res. 2011;191:155-69.
21. Wang J-B, Wang M-Z. Worst-case analysis for flow shop scheduling problems with an exponential learning effect. J Oper Res Soc. 2012;63:130-7.
22. Sun L-H, Cui K, Chen J-H, Wang J, He X-C. Research on permutation flow shop scheduling problems with general position-dependent learning effects. Ann Oper Res. 2013;211:473-80.
23. Li G, Wang X-Y, Wang J-B, Sun L-Y. Worst case analysis of flow shop scheduling problems with a time-dependent learning effect. Int J Prod Econ. 2013;142:98-104.
24. Bai D, Tang M, Zhang Z-H, Santibanez-Gonzalez EDR. Flow shop learning effect scheduling problem with release dates. Omega. 2018;78:21-38.
25. Swan J, Adriensen S, Brownlee AEI, Hammond K, Johnson CG, Kheiri A. Metaheuristics “in the large”. Eur J Oper Res. 2022;297:393-406.
26. Zhao H, Kong F. Research and applications of shop scheduling based on Genetic Algorithms. Braz Arch Biol Technol. 2016;59.
27. Martínez-Álvarez F, Assencio-Cortês G, Torres JF, Gutierrez-Avilés D, Melgar-Garcia L, Péres-Chacón R. Coronavirus Optimization Algorithm: a bioinspired metaheuristic based on the Covid-19 propagation model. Big Data. 2020;3:1-15.
28. Hosseini E, Ghafoor KZ, Sadiq AS, Guizani M, Emrouzneiad A. Covid-19 optimizer algorithm, modeling and controlling of coronavirus distribution process. IEEE J Biomed Health Inform. 2020;24:2765-75.
29. Al-Betar MA, Alyasseri ZAA, Awadallah MA, Doush IA. Coronavirus herd immunity optimizer (CHIO). Neural Comput Appl. 2021;33:5011-42.

30. Alweshah M, Alkhalaileh S, Al-Betar MA, Bakar AA. Coronavirus herd immunity optimizer with greedy crossover for feature selection in medical diagnosis. Knowl Based Syst. 2022;235:107629.
31. Fuchigami HY, Prata BA. Coronavirus optimization algorithms for minimizing earliness, tardiness, and anticipation of due dates in permutation flow shop scheduling. Arab J Sci Eng. 2023.
32. Zhao F, Qin S, Yang G, Ma W, Zhang C, Song H. A factorial based particle swarm optimization with a population adaptation mechanism for the no-wait flow shop scheduling problem with the makespan objective. Expert Syst Appl. 2019;126:41-53.
33. Tasgetiren MF, Liang Y-C, Sevkli M, Gencyilmaz G. A particle swarm optimization algorithm for makespan and total flow time minimization in the permutation flow shop sequencing problem. Eur J Oper Res. 2007;177:1930-47.
34. Krause J, Cordeiro J, Parpinelli RS, Lopes HS. A survey of swarm algorithms applied to discrete optimization problems. Swarm Intell Bio-Inspired Comput. 2013:169-91.
35. Power JM, Ou-Yang C, Juan Y-C. Optimize batch size combination using improved hybrid particle swarm optimization. Procedia Comput Sci. 2022;197:370-6.
36. Ding J, Schulz S, Shen L, Buscher U, Lü Z. Energy aware scheduling in flexible flow shops with hybrid particle swarm optimization. Comput Oper Res. 2021;125:105088.
37. Lin S, Liu A, Wang J, Kong X. An intelligence-based hybrid PSO-SA for mobile robot path planning in the warehouse. J Comput Sci. 2023;67:101938.
38. Tan Y, Zhu Y. Fireworks algorithm for optimization. Proc Int Conf Swarm Intell. 2010:355-64.
39. Li J, Tan Y. A comprehensive review of the fireworks algorithm. ACM Comput Surv. 2019;52:1-28.
40. Tan Y. Fireworks Algorithm: a novel swarm intelligence optimization method. Springer; 2015.
41. Li X-G, Han S-F, Gong C-Q. Analysis and improvement of fireworks algorithm. Algorithms. 2017;10:1-22.
42. He L, Li W, Zhang Y, Cao Y. A discrete multi-objective fireworks algorithm for flow shop scheduling with sequence-dependent setup times. Swarm Evol Comput. 2019;51:100575.
43. Pang X, Xue H, Tseng M-L, Lim M-K, Liu K. Hybrid flow shop scheduling problems using improved fireworks algorithms for permutation. Appl Sci. 2020;10:1-16.
44. Rao RV, Savsani VJ, Vakharia DP. Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput Aided Des. 2011;43(3):303-15.
45. Zou F, Chen D, Xu Q. A survey of teaching-learning-based optimization. Neurocomputing. 2019;335:366-83.
46. Xie Z, Zhang C, Shao X, Lin W, Zhu H. An effective hybrid teaching-learning-based optimization algorithm for the permutation flow shop scheduling problem. Adv Eng Softw. 2014;77:35-47.
47. Baykasoglu A, Hamzadayi A, Köse SY. Testing the performance of teaching-learning based optimization (TLBO) algorithm on combinatorial problems: flow shop and job shop scheduling cases. Inf Sci. 2014;276:204-18.
48. Shao W, Pi D, Shao Z. A hybrid discrete optimization algorithm based on teaching-probabilistic learning mechanism for no-wait flow shop scheduling. Knowl Based Syst. 2016;107:219-34.
49. Rao RV. Jaya: a simple and new optimization algorithm for solving constrained and unconstrained optimization problems. Int J Ind Eng Comput. 2016;7:19-34.
50. He L, Li W, Chiong R, Abedi M, Cao Y, Zhang Y. Optimizing the job-shop scheduling problem using a multi-objective Jaya algorithm. Appl Soft Comput. 2021;111:107654.
51. Rao RV. Jaya: an advanced optimization algorithm and its engineering applications. Springer; 2019.
52. Zhang Y, Chi A, Mirjalili S. Enhanced Jaya algorithm: a simple but efficient optimization method for constrained engineering design problems. Knowl Based Syst. 2021;233:107555.
53. Buddala R, Mahapatra SS. Improved teaching-learning-based and Jaya optimization algorithms for solving flexible flow shop scheduling problems. J Ind Eng Int. 2018;14:555-70.
54. Mishra AK, Shrivastava D. A discrete Jaya algorithm for permutation flow shop scheduling problem. Int J Ind Eng Comput. 2020;11:315-28.
55. Fan J, Shen W, Gao L, Zhang C, Zhang Z. A hybrid Jaya algorithm for solving flexible job shop scheduling problem considering multiple critic paths. J Manuf Syst. 2021;60:298-311.
56. Mishra AK, Shrivastava D, Bundela B, Sircar S. An efficient Jaya algorithm for multi objective permutation flow shop scheduling problem. Adv Intell Syst Comput. 2019;949:113-25.
57. Caldeira RH, Gnanavelbabu A. A Pareto based discrete Jaya algorithm for multi-objective flexible job shop scheduling problem. Expert Syst Appl. 2021;170:114567.
58. Abreu AP, Fuchigami HY. An efficiency and robustness analysis of warm-start mathematical models for idle and waiting times optimization in the flow shop. Comput Ind Eng. 2022;166:107976.
59. Fuchigami HY, Sarker R, Rangel S. Near-optimal heuristics for just-in-time jobs maximization in flow shop scheduling. Algorithms. 2018;11:1-17.
60. Ronconi DP, Birgin EG. Mixed-integer programming models for flowshop scheduling problems minimizing the total earliness and tardiness. In: Just-in-Time systems. Springer: New York; 2012. p. 91-105.
61. Stafford Junior EF, Tseng FT, Gupta JND. Comparative evaluation of MILP flowshop models. J Oper Res Soc. 2005;56:88-101.
62. Tseng FT, Stafford Junior EF, Gupta JND. An empirical analysis of integer programming formulations for the permutation flow shop. Omega. 2004;3:285-93.
63. Wilson JM. Alternative formulations of a flow-shop scheduling problem. J Oper Res Soc. 1989;40:395-9.
64. Taillard E. Benchmarks for basic scheduling problems. Eur J Oper Res. 1993;64(2):278-85.

65. Vallada E, Ruiz R, Framinan JM. New hard benchmark for flowshop scheduling problems minimizing makespan. Eur J Oper Res. 2015;240(3):666-77.
66. Pan QK, Ruiz R. An effective iterated greedy algorithm for the mixed no-idle permutation flowshop scheduling problem. Omega. 2014;44:41-50.
67. Montgomery DC. Design and analysis of experiments. John Wiley & Sons; 2017.
68. Dolan ED, Moré JJ. Benchmarking optimization software with performance profiles. Math Program. 2002;91(2):201-13.
69. Gould N, Scott J. A note on performance profiles for benchmarking software. ACM Trans Math Softw (TOMS). 2016;43(2):1-5.
70. Freitas MG, Fuchigami HY. A new technology implementation via mathematical modeling for the sequence-dependent setup times of industrial problems. Comput Ind Eng. 2022;172:108624.
71. Moreno A, Munari P, Alem D. A branch-and-benders-cut algorithm for the crew scheduling and routing problem in road restoration. Eur J Oper Res. 2019;275(1):16-34.