Swarm and Evolutionary Computation 75 (2022) 101149

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

# A novel BRKGA for the customer order scheduling with missing operations to minimize total tardiness

Levi Ribeiro de Abreu [a,*], Bruno de Athayde Prata [b], Allan Costa Gomes [c], Stéphanie Alencar Braga-Santos [d], Marcelo Seido Nagano [a]

[a] Department of Production Engineering, University of São Paulo, São Carlos, Brazil
[b] Department of Industrial Engineering, Federal University of Ceara, Fortaleza, Brazil
[c] Department of Electrical Engineering, Federal University of Ceara, Fortaleza, Brazil
[d] Department of Teleinformatics Engineering, Federal University of Ceara, Fortaleza, Brazil

ARTICLE INFO

*Keywords:*
Customer order scheduling
Assembly scheduling
Genetic algorithms
Missing operations
Matheuristics

ABSTRACT

We introduce a new variant of the customer order scheduling problem with missing operations to minimize total tardiness. This problem arises in the pharmaceutical industry, more specifically in physical–chemical analysis processes. Since each sample must be processed in some specific machines, we have missing operations. Given the NP-hardness of the problem, we present approximate algorithms to solve large-sized instances. First, we propose an innovative size-reduction matheuristic for a scheduling problem with due dates. This approach is based on partitioning the decision variables considering due dates and a dispatch rule. Furthermore, we develop a novel Biased Random Key Genetic Algorithm (BRKGA) that considers an efficient local search as 2-opt best improvement with swap neighborhood and a parameter-free restart procedure which restarts the search if the quality of the worst and best solutions were equal, minimizing the amount of parameters to be defined by the BRKGA. We perform computational experiments on 640 test instances to evaluate the proposed solution approaches. The results indicate the superiority of BRKGA compared to the competitive algorithms for order scheduling and its recent variants. In all set of instances, the novel BRKGA performed better than benchmarking methods and mathematical programming models, with average relative deviation index regarding best results as lower as 0.15%. Computational results point to the capacity of the proposed approaches to solve large-sized problems.

## 1. Introduction

In the last few years, the interest in assembly scheduling problems increased substantially [1]. Nowadays, companies are faced with new technologies and actuation at a global level. The customer orders are produced in several production lines and assembled in a final operation. If the processing time of the assembly operation is null, we have the customer order scheduling environment. In this context, since the products present due dates associated with the customer requirements, the minimization of total tardiness plays a key role in production planning.

Production sequencing problems in which some jobs can be skipped in some machines appear in many real-world scenarios, such as the steel industry [2]. Several researchers have reported production environments with missing operations, such as permutation flowshop [3,4], two-stage hybrid flowshop [2], multi-stage hybrid [5], cellular manufacturing [6]. Several researchers have reported production environments with energy efficient problems and soft computing techniques [7,8]. Concerning customer order scheduling, in the classical variant with total tardiness minimization, there are $m$ independent single-machine scheduling problems where each machine presents the same fixed permutation [9]. With the missing operations, there are distinct routes.

The variant under study appears in the pharmaceutical industry, more precisely in laboratories for quality control of raw materials, in-process products, and completed goods. The planning of the laboratory is based on the correct resource allocation, aiming at reducing the operational costs as well as the lead times.

We present an overview of the process in the laboratory. First, the laboratory receives samples of raw materials and finished products that must be analyzed in distinct types of equipment. The samples arrive in the laboratory, requiring a specific analysis of several types of equipment (machines). Each sample presents a given due date as well as setup time. In the following topics, we describe the main assumptions of the problem under study:

* Corresponding author.
*E-mail address:* leviribeiro@alu.ufc.br (L.R. de Abreu).

https://doi.org/10.1016/j.swevo.2022.101149
Received 30 April 2022; Received in revised form 11 July 2022; Accepted 4 August 2022
Available online 13 August 2022

- Each analysis must be processed for one machine among the set of available machines in the laboratory.
- Each machine can perform a given type of analysis.
- The processing and setup times are deterministic and previously known.
- The due dates are defined taking into consideration the urgency of each type of sample.
- The analysis for each sample not necessarily must be performed by all the available machines.
- Since a sample is analyzed in a given machine, it must be prepared for the next analysis, and this setup time depends on the type of sample.

Since the setup times are order-dependent, they can be added to the processing times. In this variant, some orders are not processed on all machines. Thereby, we observe a new variant of the customer order scheduling considering missing operations to minimize the total tardiness. A set of orders must be processed on a set of machines with specific processing times, and each order does not need to be processed on all machines. Therefore, there are missing operations. The new problem has the form $DPm \rightarrow 0|missing|\Sigma T$ in the notation proposed by Framinan et al. [1].

In this paper, our main contributions are listed below:

- We introduce the customer order scheduling with missing operations.
- We develop two analytical models: mixed-integer linear programming (MILP) formulation, and a constraint programming (CP) model.
- We propose a new matheuristic named size reduction with partitions (SR-Q) which is developed for scheduling problems with due dates.
- We develop an innovative BRKGA which considers a robust improvement procedure and a parameter-free restart operator.
- We perform computational experiments with a testbed with 640 test instances.

In this paper, we present a customer order scheduling environment with missing operations and total tardiness minimization. We propose a size-reduction heuristic and a BRKGA that outperforms the JPO matheuristic, Framinan and Perez-Gonzalez [10] and adapted to the problem under study, for the large-sized test instances. Finally, we perform extensive computational experiments.

The main contributions and innovations of the new BRKGA are the application of a local search 2-opt best improvement procedure as a search intensification mechanism that considers the swap neighborhood of the problem and the restart mechanism with parameter-free to restart the search process if there are no more improvements in the BRKGA loop when the quality of the best individual are equal to the worst individual of the population, thus reducing the amount of BRKGA parameters and simplifying the parameterization of the metaheuristic.

The remainder of the paper is structured as follows. Section 2 presents the literature review. Section 3 addresses the problem statement as well as the MILP and CP models. In Section 4, we propose an innovative size-reduction matheuristic with partitions. In Section 5 we describe the computational results. Finally, in Section 6 we present the main finding ans suggestion for future studies.

## 2. Literature review

The production environment addressed in this paper was not reported yet. In this context, we present some related approaches. Sung and Yoon [11] studied a variant where the orders have distinct components produced by specialized and independent machines. The objective function is the weighted total completion time minimization. As solution procedures, two constructive heuristics are developed. Ahmadi et al. [12] addressed the coordinated customer order scheduling with weighted completion time minimization. A Lagrangian heuristic and three constructive heuristics are presented as solution procedures. The heuristics developed by Leung et al. [13] improved the results of the solution approaches previously addressed. Still regarding the weighted total completion time objective, Wu et al. [14] introduced a variant with ready times. Some dominance relation are considered in a Branch-and-Bound (B&B) algorithm. Furthermore, five constructive heuristic are adapted to the problem and study and used as initial solutions for a Iterated Greedy (IG) algorithm.

Concerning the total completion time objective, Shi et al. [15] developed quadratic modeling, which is transformed into an equivalent MILP formulation. A nested partition algorithm is proposed as a solution approach. Framinan and Perez-Gonzalez [16] presented a constructive heuristic and a metaheuristic as solution procedures that outperform the existing algorithms for this variant. Riahi et al. [17] improved the constructive heuristic developed by Framinan and Perez-Gonzalez [16]. This algorithm is an initial solution for a metaheuristic that combines perturbative and constructive procedures.

Kung et al. [18] studied a variant considering unequal order ready times for the total completion time minimization. As solution procedures, eight metaheuristic are developed. Computational experiments point to the superiority of simulated annealing-based algorithms in comparison with genetic algorithms. With respect to the minimization of the weighted number of tardy orders objective, Lin et al. [19] studied a variant with release dates. As theoretical results, some dominance rules and a lower bound are presented. As solution procedures, a branch-and-bound (B&B) algorithm and with bee colony metaheuristic are developed.

Concerning the total tardiness minimization, Lee [9] proposed four constructive algorithms. Among them, the order modified due date (OMDD) returned the best results. Framinan and Perez-Gonzalez [10] proposed the FP algorithm that outperforms the OMDD heuristic. In addition, two matheuristics (JPF and JPO) are introduced for the problem under study. Computational results point to the superiority of the JPO algorithm.

More recently, some studies have included sequence-dependent setup times in the customer order scheduling environment. Prata et al. [20] introduced the variant for the makespan minimization. Since this is an NP-hard problem, two matheuristics are presented as solution approaches. Prata et al. [21] addressed the total completion time objective. As the solution approach, a discrete differential evolution metaheuristic was proposed. Computational results pointed out the superiority of this algorithm in comparison with other existing solution procedures. Antonioli et al. [22] considered total tardiness minimization. Two constructive heuristics were proposed, as well as a hybrid matheuristic based on the JPO algorithm [10].

Based on the revised literature, the following research gaps can be emphasized:

- The customer order scheduling problem with missing operations was not addressed yet in the current literature.
- The total tardiness performance measure is a topic little studied in the customer order scheduling environment.
- The development of a hybrid metaheuristic for a customer order scheduling problem is still rather limited in the available literature.

## 3. Problem description and exact methods

The problem under study considers a set of orders, with some missing operations and order-dependent setup times to be added with the process times and processed in a set of dedicated parallel machines. The objective function is the total tardiness minimization. Let a set of $m$ dedicated parallel machines and a set of $k$ orders to be fulfilled, in which each order presents a due date $d_k$. In each machine, a given order presents an associated processing time $p_{ik}$ for each existing operation.

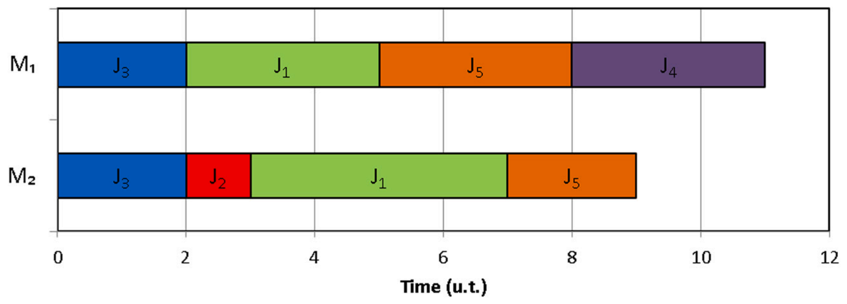

**Fig. 1.** Gantt chart of an example solution for presented order scheduling instance (total tardiness = 11 u.t.).

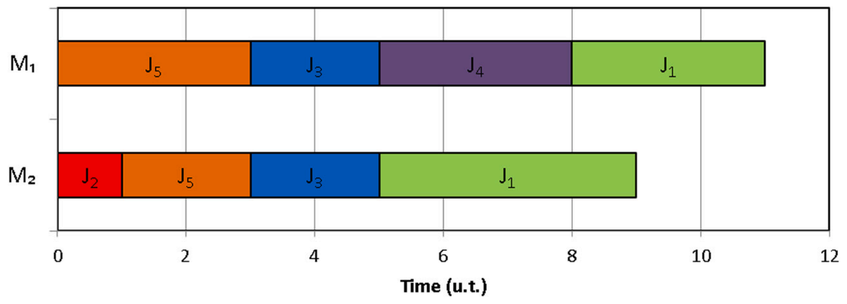

**Fig. 2.** Gantt chart of the best solution for presented order scheduling instance (total tardiness = 7 u.t.).

**Table 1**
Operations and processing times for order scheduling with missing operations example.

| $p_{ij}$ | $O_1$ | $O_2$ | $O_3$ | $O_4$ | $O_5$ |
|---|---|---|---|---|---|
| $M_1$ | 3 | – | 2 | 3 | 3 |
| $M_2$ | 4 | 1 | 2 | – | 2 |
| $d_k$ | 4 | 5 | 6 | 8 | 4 |

For representing the missing operations, we use a boolean parameter $a_{ik}$ equals to one if the order $k$ is processed in machine $i$, being zero otherwise and $\Theta$ is the percentage of the processing time matrix with missing operations.

A feasible solution for the problem is a permutation $\Pi = \{\pi_1, \dots, \pi_n\}$ in which $\pi_j \in \Pi$ is the order to be produced in the position $j$ of the production sequence. The completion time $C_{i\pi_j}$ of the operation from order $\pi_j$ processed in position $j$ of machine $i$ is given by Eq. (1):

$$C_{i\pi_j} = \sum_{q=1}^{j} p_{i\pi_q} a_{i\pi_q} \tag{1}$$

We can define the completion time $C_{\pi_j}$ for the order $\pi_j$ in position $j$ of sequence $\Pi$ as the maximum completion time among all the realized operations, as expressed by Eq. (2):

$$C_{\pi_j} = \max_{1\leq i\leq m} \{C_{i\pi_j}\} \tag{2}$$

Thus, we can compute the tardiness $T_{\pi_j}$ for each order based on Eq. (3):

$$T_{\pi_j} = \max\{C_{\pi_j} - d\{\pi_j\}; 0\} \tag{3}$$

Table 1 presents an illustrative example with $m = 2$ and $n = 5$.

Fig. 1 describes a Gantt chart with the example solution of the instance illustrated in Table 1. The solution is defined as the sequence of orders on each machine $\Pi = \{3, 2, 1, 5, 4\}$. Each machine has the same sequence of orders. The solution has a total tardiness of 11 time units (u.t.).

The solution in Fig. 1 has a total tardiness of 11 u.t. Orders 1, 5, and 4 have tardiness of 3, 5, and 3 u.t., respectively. Fig. 2 illustrates the optimal solution for the instance, with the smallest total tardiness of 7 u.t. For the solution in Fig. 2, some orders finish in sync or before their due dates, which contributes to the reduction of the total tardiness, such as orders 2, 5, 3 and 4 that do not generate delays in their processing, unlike the solution in Fig. 1 that only order 3 and 2 does not finish late.

We are investigating the generation of production sequences that minimizes the total tardiness with missing operations. This is an NP-hard problem since it can be reduced to the $DPm \rightarrow 0||\Sigma T_j$ if the number of missing operations is zero. Hereafter, the notation used for the problem modeling is presented.

## *3.1. Mixed-integer linear programming model*

Below we presented a new mixed-integer linear programming model for the order scheduling with missing operations.

**Indices and sets**

$k$: index for orders $\{1,2, \dots, n\}$.

$j$: index for positions $\{1,2, \dots, n\}$.

$i$: index for machines $\{1,2, \dots, m\}$.

**Parameters**

$p_{ik}$: processing time of the order $k$ in the machine $i$.

$$a_{ik} = \begin{cases} 1, & \text{if the order } k \text{ is produced in machine } i \\ 0, & \text{otherwise} \end{cases}$$

**Decision variables**

$T_j$: tardiness of order in the position $j$.

$$x_{kj} = \begin{cases} 1, & \text{if the order } k \text{ is scheduled in position } j \\ 0, & \text{otherwise} \end{cases}$$

The proposed mixed-integer programming model is presented as follows.

minimize

$$\sum_{j=1}^{n} T_j \tag{4}$$

subject to

$$\sum_{k=1}^{n} x_{kj} = 1, \qquad \forall j \tag{5}$$

$$\sum_{j=1}^{n} x_{kj} = 1, \qquad \forall k \tag{6}$$

$$T_j \geq \sum_{k=1}^{n} \left( \sum_{r=1}^{j} p_{ik} a_{ik} x_{kr} - d_k x_{kj} \right), \qquad \forall i, j \tag{7}$$

$$T_j \geq 0, \qquad \forall j \tag{8}$$

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

Eq. (4) is the total tardiness to be minimized. Constraints (5) establish that a given order is processed only in a position $k$. Constraints (6) enforce that a position $k$ receives only a job $j$. Constraints (7) determine the tardiness for each order, considering processing and setup times, as well as the missing operations. Finally, constraint sets (8), and (9) determine the scope of the decision variables. The proposed model presents $n^2$ binary decision variables, $n$ continuous decision variables and $n(m + 3)$ integer linear constraints.

## 3.2. Constraint programming model

Constraint programming is a modeling approach for combinatorial optimization problems. Mainly for problems that are not easily represented using integer linear programming equations [23]. Constraint programming was initially proposed for artificial intelligence problems. However, it has obtained competitive results in production, operations, and project scheduling problems [24].

The order scheduling problem can be defined using constraint programming equations and the same indexes, sets, and parameters in the presented linear programming model. With the CP model, the problem can be represented with interval and discrete decision variables. Interval variables represent an operation or task to be processed. In addition, constraint programming uses integer variables similar to MILP. The present project uses IBM's CP Optimize solver for modeling and solving the CP model. This solver has obtained good results on recent production scheduling problems, such as general scheduling [25,26], parallel machines [27,28], flow shop [29], job shop [30,31], and open shop [32,33].

Next, we illustrate a new CP model for order scheduling considering missing operations. This CP model uses intervals and variables to represent the operations of jobs in machines, integer variables to represent completion time and tardiness of orders, and the same indexes and parameters of the MILP model.

**Decision variables**

$x_{ik}$: an interval variable to indicate the operation of order $k$ in machine $i$ with a duration $p_{ik}$ just of operations with non-zero processing times ($a_{ik} = 1$).

$C_k$: completion time of order $k$.

$T_k$: tardiness of order $k$.

The CP model for the problem under study is presented below:

minimize

$$\sum_{k=1}^{n} T_k \tag{10}$$

**Table 2**
Resume of comparison of the models.

| Model | Number of IVs | Number of CVs | Number of constraints |
|---|---|---|---|
| MILP | $n^2$ | $n$ | $n^2 + n$ |
| CP | $mn + 2n$* | N/A | $mn + m + n$ |

subject to

$$\text{noOverlap}\left(\left[x_{ik}\right]_{k, a_{ik}=1}\right), \qquad \forall i \tag{11}$$

$$C_k \geq \text{endOf}\left(x_{ik}\right), \qquad \forall i, k, a_{ik} = 1 \tag{12}$$

$$T_k \geq C_k - d_k, \qquad \forall k \tag{13}$$

$$\text{interval } x_{ik}, \quad \text{size} = p_{ik}, \qquad \forall i, k, a_{ik} = 1 \tag{14}$$

$$C_k \geq 0, \qquad \forall k \tag{15}$$

$$T_k \geq 0, \qquad \forall k \tag{16}$$

Eq. (10) is the total tardiness minimization. Constraint set (11) enforces that a machine $i$ processes only one order at a time (overlap constraints). Constraint set (12) calculates the completion time of orders with the maximum completion time of all operations of order $k$ in all machines. Constraint set (13) calculates the tardiness of order $k$. Finally, constraints (14), (15), and (16) define the scope of decision variables.

We compare the main components of order scheduling models with MILP and CP. MILP model uses positional notation for decision variables. The constraint with the largest size is the one found in Eq. (7) with the worst case complexity of $O(mn)$. The MILP model has altogether $2n + mn$ constraints and $n^2 + n$ decision variables, of which $n$ are real and $n^2$ are binary integers [34]. The proposed CP model applies heuristic approaches in logical constraints of the problem to reduce the search space and improve the branching strategy of the solver. Moreover, the CP solver keeps an ability to find feasible initial solutions quickly due to explorations of the combinatorial domain of the problem [23]. Using logical constraints, the CP model can significantly reduce the number of decision variables and constraints, improving the performance of combinatorial production scheduling problems.

We compare the main components of order scheduling models with MILP and CP. MILP model uses positional notation for decision variables. The constraint with the largest size is the one found in Eq. (7) with the worst case complexity of $O(mn)$. The MILP model has altogether $2n + mn$ constraints and $n^2 + n$ decision variables, of which $n$ are real and $n^2$ are binary integers [34]. The CP model uses logical modeling and applies heuristic approaches to reduce the search space for the problem solution. In addition, exploring the combinatorial search space of the problem makes it very fast for CP to find a viable solution [23]. This result is very important when modeling combinatorial problems such as production scheduling, as it may mainly use logical modeling techniques. This reduces the number of decision variables and model constraints.

The new CP model is presented in Eqs. (10)–(16). We used the sequence problems notation of the CP Optimizer solver. Eq. (12) illustrates the constraint with the largest size with the complexity of only $O(nm)$ in the worst case. The proposed CP model is formed by $mn+m+n$ constraints and $mn + 2n$ decision variables. The decision variables have discrete types such as interval or integer decision variables. Table 2 shows a comparison of models with the number of integers or discrete variables (IVs), continuous variables (CVs), and constraints (* the decision variables of CP are discrete and their types are interval or integer types). Analyzing Table 2, the two proposed exact models have similar complexity of constraints and decision variables. However, similar models will not necessarily have the same performance, therefore all exact methods will be performed in Section 5.4.

```
Algorithm SR-$Q$ {$\mathbf{x}$, $\mathbf{d}$, $Q$};
sort the indeces $\mathbf{I}_d$ of $\mathbf{d}$ in non decreasing order;
split $\mathbf{I}_d$ in $Q$ partitions;
for each partition $q \in \{1, ..., Q\}$ do
    for all orders $k$ in $q$ do
        for all the other partitions $r \in (\{1, ..., Q\} - \{q\})$ do
            for all orders $j$ in $r$ do
                fix $x_{kj} = 0$;
            end
        end
    end
end
return {$\mathbf{x}$};
```

**Fig. 3.** Pseudocode of the algorithm SR-$Q$.

**Table 3**
Illustration of a SR-2 variable fixing scheme.

| x | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | – | – | – | – | – | 0 | 0 | 0 | 0 | 0 |
| 2 | – | – | – | – | – | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | – | – | – | – | – |
| 4 | 0 | 0 | 0 | 0 | 0 | – | – | – | – | – |
| 5 | – | – | – | – | – | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | – | – | – | – | – |
| 7 | – | – | – | – | – | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | – | – | – | – | – |
| 9 | – | – | – | – | – | 0 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 | 0 | – | – | – | – | – |

**Table 4**
Illustration of a SR-3 variable fixing scheme.

| x | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | – | – | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | – | – | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | – | – | – | – | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – |
| 5 | 0 | 0 | 0 | – | – | – | – | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – |
| 7 | – | – | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | – | – | – | – | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | – | – | – | – | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – |

## 4. Proposed solution approaches

The following subsections describe the new proposed matheuristic SR-Q and metaheuristic BRKGA with local search and parameter-free restart procedure for the order scheduling problem with missing operations.

### 4.1. Size reduction with partitions

In combinatorial optimization problems with permutation encoding, we usually face binary decision variables in which the majority of the decision variables values with null values in a feasible solution. In this situation, we could reduce the number of decision variables, fixing a subset of the total number of decision variables as zero. Thus, we can solve the model with this reduction with a lesser computational effort [35]. However, we cannot guarantee that the optimal solution of the reduced problem is the optimal solution for the original problem.

Fanjul-Peyro and Ruiz [36] introduced the size reduction heuristic for the unrelated parallel machine scheduling problem to minimize the makespan. In this approach, the decision variables associated with the highest processing times are set as zero, and the model with the remainder free decision variables is solved.

Several studies have presented matheuristics as extensions and improvements of MILP models in production scheduling problems [20,37,38]. The SR-Q matheuristic is a natural extension of the proposed MILP model, along with the knowledge of the problem properties concerning the matrix form of the decisions variables. The intuitive property of the problem used is that orders with shorter deadlines should be processed first in the solution. Benchmarking analysis will test this matheuristic as another competitive alternative to the problem.

For the problem under study, the objective function is the total tardiness minimization; therefore, we propose a new size-reduction strategy based on the due dates. In this innovative procedure, we split the set of decision variables on the basis of due dates. In Fig. 3, we describe the algorithm size-reduction with partitions (SR-$Q$). The values of some decision variables $x_{kj}$ are fixed as zero, reducing the number of decision variables in the optimization process. With a reduction of the number of decision variables, in general, there is also a reduction in, the computational effort for the problem resolution.

Since the objective function is the total tardiness minimization, we can use the due dates as a criterion for the size-reduction. Intuitively, orders with small due dates should be scheduled in the first positions of the sequence. Similarly, orders with large due dates should not be scheduled in the first positions of the sequence.

If we are looking for the total tardiness minimization, orders with small due dates tend to appear in the first positions of the sequence. Similarly, orders with large due dates tend to appear in the last positions of the sequence. This information is the key feature of SR-Q compared to other traditional exact methods, using the knowledge of the problem due date to fix the variables and consequently reduce the number of decision variables in the MILP model. Therefore, we can exclude the possibility that orders with tight due dates appear at the beginning of the sequence, as well as orders with large due dates, appear at the ending of the sequence.

On the basis of the above, we develop the following strategy for tackling the problem: the orders are divided into $Q$ mutually exclusive partitions taking into consideration the due dates. We set $x_{kj} = 0$ is the order $k$ does not belong to the partition related to the position $j$.

We present an illustrative example with 10 orders and a due dates vector $d = \{3, 2, 5, 7, 4, 9, 2, 5, 4, 8\}$. For $Q = 2$, there are two partitions $Q_1 = \{2, 7, 1, 5, 9\}$ and $Q_2 = \{3, 8, 4, 10, 6\}$. We can observe that these partitions are divided taking into account the indices of the orders. The next step is the variable fixing of the decision variables $x_{kj}$, as presented in Table 3.

Taking into consideration the same instance, for $Q = 3$ we adopt as the amount of elements $N_q$ in each partition the rounding of the ratio $qn/Q$ minus the amount of elements in the previous partitions, as expressed by Eq. (17), in which $q \in \{1, \dots, Q\}$ is the partition $q$. In the above mentioned example, we have $N_1 = 3$, $N_2 = 4$, and $N_3 = 3$. Thus, we obtain the following partitions $Q_1 = \{2, 7, 19\}$, $Q_2 = \{5, 9, 3, 8\}$, and $Q_2 = \{4, 10, 6\}$. Table 4 illustrates the values for the decision variables $x_{kj}$.

$$N_q = \lceil (qn/Q) \rceil - \sum_{r=1}^{q-1} N_r \tag{17}$$

Based on the Tables 3 and 4, we can observe that the application of the algorithms SR-2 and SR-3 results in a reduction of the number of binary decision variables $x_{kj}$ by half and third part, respectively. Therefore, the application of size-reduction with $Q$ leads to a reduction of the binary decision variables from de $n^2$ para $n^2/q$.

### 4.2. Biased random-key genetic algorithm

The BRKGA is an evolutionary metaheuristic based on the core concepts of a genetic algorithm in which each solution of a given population has a vector with random keys [39]. Thereby, the crossover operator is performed in a biased way, favoring the genetic material of the better solutions in the current population. Some of these new

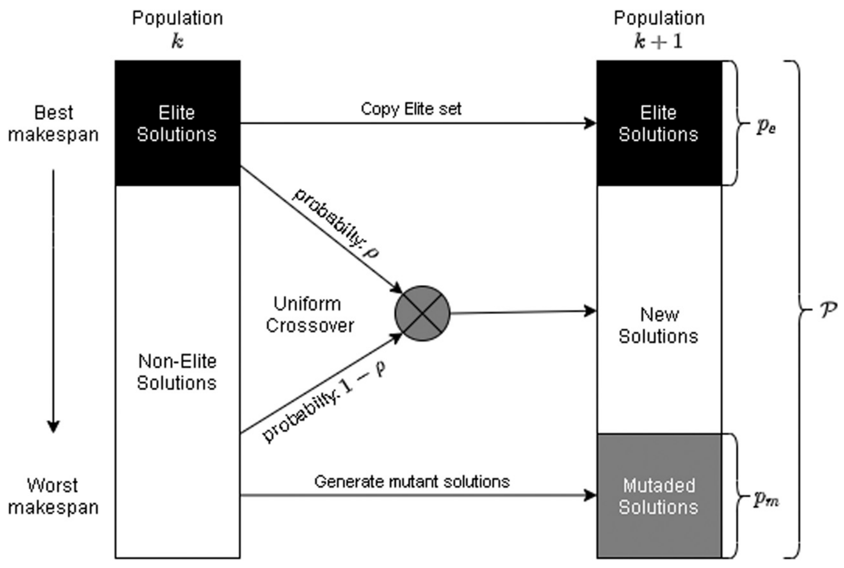

**Fig. 4.** Scheme for the main process of population evolution of BRKGA.

| Sequence representation: | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Key solution representation: | 0.67 | 0.92 | 0.15 | 0.45 | 0.05 |
| Decoded solution: | 5 | 3 | 4 | 1 | 2 |

**Fig. 5.** Proposed decoder for the order scheduling problem.

operators of genetic algorithms and BRKGA were applied in other metaheuristics as differential evolution [40,41]. The BRKGA was initially proposed to solve problems with a permutation encoding where the solution can be structured as a sequence of values [42]. With this structure, the BRKGA has been widely used in resolving production scheduling problems, such as parallel machines [43–46], flow shop [47–50], job shop [51–53], and open shop [54–56].

In our proposal, the BRKGA generates an initial population with $p$ random keys with float values between 0 and 1. A single element of this initial population is replaced with a solution generated by the well-known earliest due date (EDD) dispatch rule. The solution created with the EDD is coded with random keys to be included as a valid solution in the initial population.

Next, the BRKGA initiates its cycle of generations. Fig. 4 illustrates the scheme for the evolution process of population $\mathcal{P}$. In this process, a decoding function converts the key vector in the total tardiness of a given solution. Thus, all solutions in the current population are sorted considering the values of the objective function. The number of $p_e$ better solutions is included in the elite set; thereby, they are maintained in the next generation.

The $p_m$ worse solutions are replaced with new mutant solutions generated using the same procedure to generate the initial population. The remainder of the new population is inserted using crossovers $p - p_e - p_m$ between a member of the elite set and a solution outside the elite set. We employ the uniform crossover in which the probability of insertion of gen from a given parent is given by $\rho$. These steps are repeated during a given number of generations or a specified time limit. In our proposal, only the decoding operator is problem-dependent. It must be designed to convert the characteristics of a given solution to the format of random keys. Thus, we can calculate the objective function value for any decoded solution. Fig. 5 illustrates the proposed decoding scheme.

Fig. 5 illustrates a decoded solution for an instance with five orders, generation the solution $\{5,3,4,1,2\}$ in which the indices represent each of the keys. This permutation represents the global optimal solution of the illustrative instance presented in Section 3. Moreover, the proposed decoding operator does not generate infeasible solutions for the problem under study, and the objective function can be easily calculated with Eq. (1), (1), and (3).

Aiming to make the proposed BRKGA more robust, we perform two strategies of diversification and intensification. For the diversification, we adopt a restart procedure based on the objective function of the solutions of the current population. If the worst fitness equals the best fitness, we restart the current population. Thus, we remove the parameter for the restart procedure, and we reset the population in the exact iteration in which the current population lost all diversity. This strategy for the restart procedure has obtained competitive results for another customer order scheduling environment [21]. Concerning the intensification, we perform every $L$ iterations of the BRKGA a 2-opt local search with the best improvement policy in the best solution of the elite set. The swap neighborhood operator is used since the solution is a list with the order sequence, and a swap neighborhood is a natural approach for this type of solution representation used in many production scheduling problems [48]. Fig. 6 describes the pseudocode of the proposed BRKGA. Our proposal presents the following parameters: $p$, $p_e$, $p_m$, $\rho$, $L$, $gen$ e $time\_limit$. In Section 5 we propose a parameter's optimization process for BRKGA with Iterated Race for Automatic Algorithm Configuration (IRACE) package [57].

In lines 1–9, the data structures and parameters of BRKGA are defined. In line 10, the loop of the algorithm starts. Then, the algorithm executes its genetic operators until the execution time or the number of generations has not exceeded the limit. In line 11, the BRKGA generation phases and operations are executed the same way as in Fig. 4. In each iteration, one population evolved with crossover and mutation operators between the elite and non-elite solutions to create a new population for the next algorithm iteration. In lines 12–15, the algorithm performs in each $L$ generation the 2-opt best improvement local search procedure in the best solution found of population $\mathcal{P}$. The local search (line 14) uses swap neighborhood in operations list solution representation as in Fig. 5. In lines 16–22, the BRKGA tests if the generation found no better solution than $bestsol$. If the best and worse finesses of the current population are the same, the algorithm executes the restart in line 18. Otherwise, the best solution is updated. Finally, in line 25, the best solution found is returned.

## 5. Computational experience

This section illustrates: the process for constructing the instance set, the performance measures, the parameter's optimization process

```
Result: A $bestsol$ solution found
1  $p \leftarrow \alpha \times n$ ;                // number of individuals in population $\mathcal{P}$
2  $p_e \leftarrow$ percentage of elite set;
3  $p_m \leftarrow$ percentage of mutant set;
4  $\rho \leftarrow$ probability to select a elite father key in crossover;
5  $L \leftarrow$ number of generations to apply local search;
6  $gen \leftarrow \beta \times n$ ;                            // number of generation
7  Initialize population $\mathcal{P}$ with $p$ random individuals and EDD solutions;
8  $bestsol \leftarrow$ EDD solution found;
9  $iter \leftarrow 1$;
10 while $iter < gen$ or $run\ time\ does\ not\ end\ (time_limit)$ do
11     Evolve population $\mathcal{P}$ just one generation ;  // Using $\mathcal{P}$, $p$, $\rho$, $p_e$ and $p_m$
12     $sol \leftarrow$ best solution $\in \mathcal{P}$;
13     if $iter \mod L = 0$ then
14         Apply the local search in $sol$ and inject chromosome in population $\mathcal{P}$;
15     end
16     if $makespan(sol) \geq makespan(bestsol)$ then
17         if $best\ fitness = worst\ fitness \in \mathcal{P}$ then
18             Replace population $\mathcal{P}$ with $bestsol$, EDD solution and random
                individuals;
19         end
20     else
21         $bestsol \leftarrow sol$;
22     end
23     $iter \leftarrow iter + 1$
24 end
25 return $bestsol$
```

**Fig. 6.** Pseudocode of the proposed BRKGA.

of BRKGA, the description of benchmarking algorithms, the computational results, and statistical tests performed.

We implement all the constructive heuristics as well as the matheuristics using Julia language (https://julialang.org/) within Atom IDE (https://atom.io/). For the pure MILP and CP model as well as the matheuristics the commercial solver is the IBM ILOG CPLEX for MILP and CP Optimizer for CP (https://www.ibm.com/products/ilog-cplex-optimization-studio) version 12.8. We perform the computational experience on a PC with Intel Core i7-8700 CPU 3.20 GHz and 32 GB memory. All instances and results are available at https://doi.org/10.13140/RG.2.2.23905.17766/1.

### 5.1. Test instances and statistics used in the computational experiments

Since the problem under study is not reported in the previous literature, we generate a set of test instances for the evaluation of the integer linear programming as well as the presented heuristics. We adapt the procedures of Lee [9] and Framinan and Perez-Gonzalez [10] for the random generation of the test instances:

- the processing times $p_{ik}$ follow a uniform distribution $U[1, 100]$;
- the due dates $d_k$ follow a uniform distribution $U[P(1 - TF - RDD/2), P(1 - TF + RDD/2)]$, in which $P$ is given by Eq. (18), and $TF$ and $RDD$ are factors for controlling the interval for the variations of the due dates.

$$P = \frac{\sum_{i=1}^{m} \sum_{k=1}^{n} p_{ik} a_{ik}}{m} \tag{18}$$

- the missing operations matrix $a_{ik}$ present a density $\Theta$ of missing operations and is also guaranteed that there is at least one operation for each order (in other words, $\sum_{i=1}^{m} a_{ik} \geq 1,\ \forall k$)

For each level of parameters ($n \in \{100, 150, 200, 300\}$, $m \in \{5, 10\}$, $TF \in \{0.35, 0.65\}$, $RDD \in \{0.35, 0.65\}$, and $\Theta \in \{60\%, 80\%\}$), a set of 10 instances are randomly generated, totaling 640 instances.

In order to compare the methods implemented, we analyze the results obtained in the computational experiments with Relative Deviation Index (RDI), as expressed in Eq. (19). The indicator RDI analyzes the relative percentage deviation taking into account all the considered methods. In Eq. (19), $T$ is the tardiness returned for a given method, $T_{best}$ is the best tardiness found for all the considered methods, and $T_{worst}$ is the worst tardiness found for all the considered methods. Framinan and Perez-Gonzalez [10] use this statistic for the $DPm \rightarrow 0||\Sigma T$ variant. For the cases in which one of the considered methods is not able to find a feasible solution in one of the sets of instances, the results of the whole set are removed from the analysis.

$$RDI = \begin{cases} 0 & if\ T_{best} = T_{worst} \\ 100 \times \frac{T - T_{best}}{T_{worst} - T_{best}} & otherwise \end{cases} \tag{19}$$

Another performance indicator used in the result analysis is the success rate (SR) that analyzed the overall performance of the methods across all tested instances. SR is calculated as the number of times that a given method finds the best solution (with or without a draw) divided by the number of test instances in a given instance set [58,59]. The SR indicator is expressed in Eq. (20), where $N_{BEST}$ is the number of times that given method finds the best solution and $N_{INST}$ is the total number of instances tested.

$$SR = 100 \times \frac{N_{BEST}}{N_{INST}} \tag{20}$$

### 5.2. Parameter's optimization of BRKGA

The IRACE package [57] has been widely used to find the best parameters for optimization algorithms like metaheuristics or matheuristics [37,60,61]. Thus, to get the most competitive parameters in the proposed SR and BRKGA algorithms, the IRACE package was applied. Table 5 shows the proposed values of each parameter. The Selected Value column shows the best value obtained by IRACE, ready to be used in the final tests. IRACE performed several runs of the algorithms with the range of possible values for each parameter. The parameter $Q$ is the partitions number of fixing decision variables and the other parameters are from BRKGA. We performed 10000 iterations of IRACE using 25% of the evaluated test instances, which were randomly selected.

In the IRACE execution, the algorithm changes the procedure the parameters are sampled adaptively to explore a range of parameter

**Table 5**
IRACE parameter range settings and resulting values.

| Parameter | IRACE name | Range | Selected value |
|---|---|---|---|
| $Q$ | Q | $\{2, 3, 5, 10\}$ | 2 |
| $\alpha$ | alpha | $\{5, 10, 20\}$ | 20 |
| $\beta$ | beta | $\{5, 15, 25\}$ | 5 |
| $p_e$ | pe | $\{0.05, 0.15, 0.25\}$ | 0.25 |
| $p_m$ | pm | $\{0.15, 0.20, 0.30\}$ | 0.15 |
| $\rho$ | rho | $\{0.35, 0.55, 0.75\}$ | 0.55 |
| $L$ | L | $\{10, 50, 100\}$ | 50 |

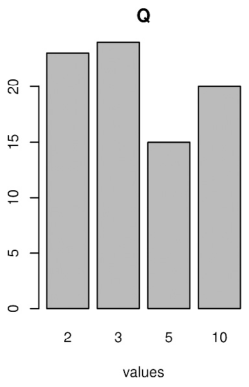

**Fig. 7.** Parameters sampling frequency for SR.

values that improve the quality of the solutions generated by the optimization algorithms. Therefore, the number of times IRACE selects each possible parameter value can indicate promising regions and explain the general behavior of the optimization algorithm when its parameter values are modified [57]. Figs. 7 and 8 show the sampling frequency of the values of each parameter for the developed methods SR and BRKGA, respectively.

As BRKGA has several new parameters, it is essential to verify the performance of each parameter with the range of values available regarding the quality of the solutions. Therefore, the test results of the IRACE parameters were collected, and the RDI from the best and worst solutions found for each instance used in the calibration was calculated. Fig. 9 illustrates the results found for each of the six BRKGA parameters.

In order to analyze the effectiveness of the local search and restart mechanisms presented in Section 4.2, we have compared four variants of BRKGA: (i) BRKGA1 (the default BRKGA initially proposed by Gonçalves and Resende [39]); (ii) BRKGA1 (the BRKGA with the proposed local search); (iii) BRKGA3 (the BRKGA with the proposed parameter-free restart procedure); and (iv) BRKGA (the proposed BRKGA with local search and restart procedures). In our tests, we used the IRACE proposed parameters in each BRKGA. Fig. 10 illustrates a boxplot with the RDI distribution of all BRKGA tested at all instance sizes. The tests ran on a randomly selected subset of 25% of the data set.

Analyzing Fig. 10, the BRKGA with local search and parameter-free restart procedure got the best results, followed by BRKGA3, BRKGA2, and BRKGA1. In addition, we performed an ANOVA test to evaluate if the difference in the RDI values was statistically significant. The $p$-value is very close to zero. Furthermore, we performed a Tukey's test [62] to evaluate the statistical significance of the differences among the BRKGA variants and show the confidence intervals in Fig. 11.

As it can be seen in both Figs. 10 and 11, it is the addition of the specific local search and parameter-free restart mechanisms that produce a significant improvement in the performance of the BRKGA, particularly when both mechanisms are combined. Thus, Fig. 11 illustrates that the BRKGA outperforms the classic BRKGA (the interval does not cross the zero line) due to the new components improving the BRKGA efficiency. In addition, the BRKGA reduces the number of parameters to be parameterized with the new parameter-free restart procedure. Given these results, we adopt BRKGA (the BRKGA variant including both local search and restart mechanisms) in the subsequent computational experiments.

### 5.3. Methods under comparison

For these tests we compare the proposed methods SR and BRKGA with benchmarking methods from the literature of order scheduling and its variants adapted for the proposed variant by us. We adopt $\frac{m \times n}{2}$ seconds as a time limit of each instance for each tested metaheuristic [21]. We adopted the same stopping criteria for all algorithms for a fair comparison. With this time limit, We can test if the approximation methods obtain quality solutions with a competitive computational times. Each of the metaheuristics tested, due to stochastic behavior, tested were executed five times, the average value found for each instance set is reported in the results.

We consider the following algorithms in our computational experiments:

- EDD, earliest due date priority rule;
- FP algorithm, presented by Framinan and Perez-Gonzalez [10];
- NEH, a constructive algorithm from Prata et al. [21] adapted to proposed variant;
- OMDD algorithm, presented by Lee [9];
- MILP model, presented in Section 3;
- CP model, presented in Section 3;

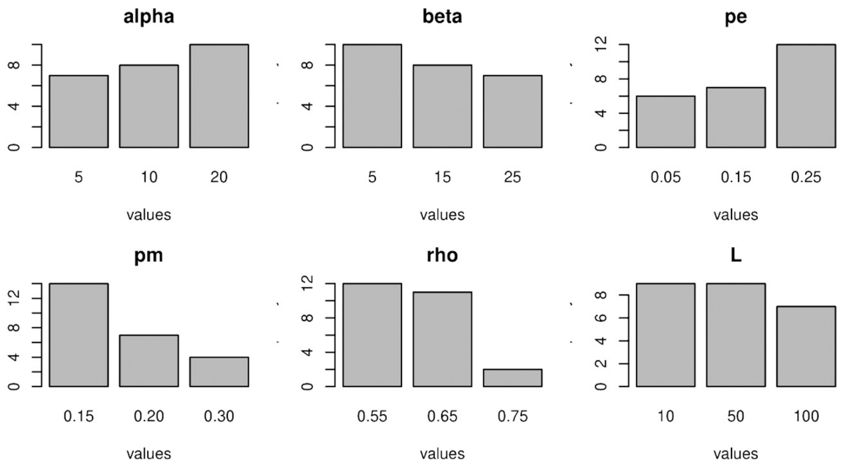

**Fig. 8.** Parameters sampling frequency for BRKGA.

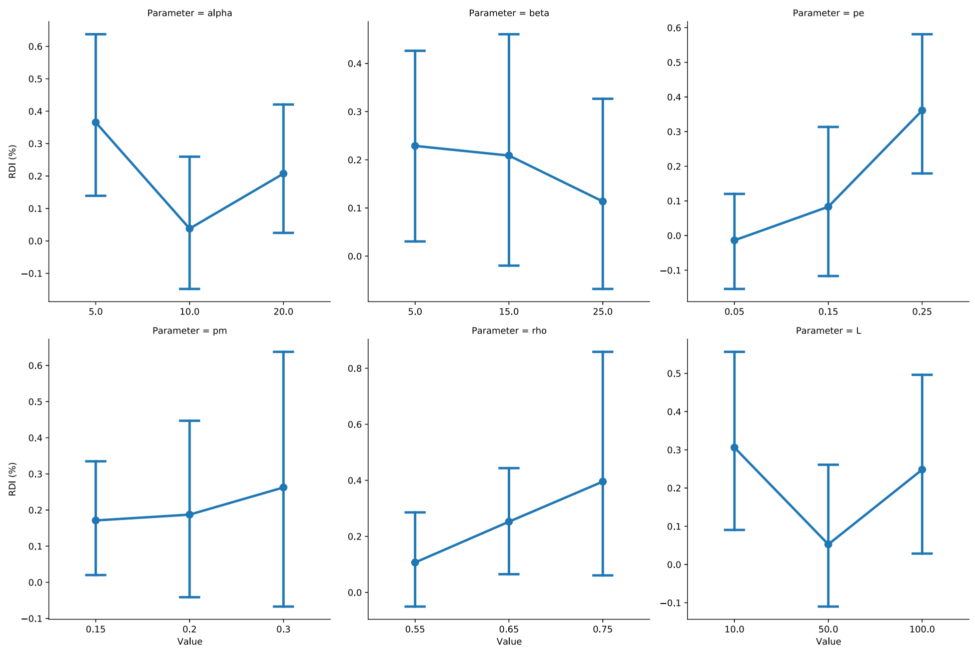

**Fig. 9.** Mean and confidence interval ($\alpha = 0.05$) of RDI for each BRKGA parameter.

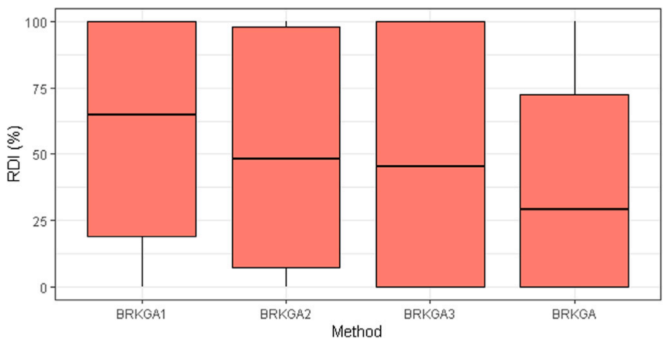

**Fig. 10.** RDI distribution for all BRKGA tested in all sets of instances.

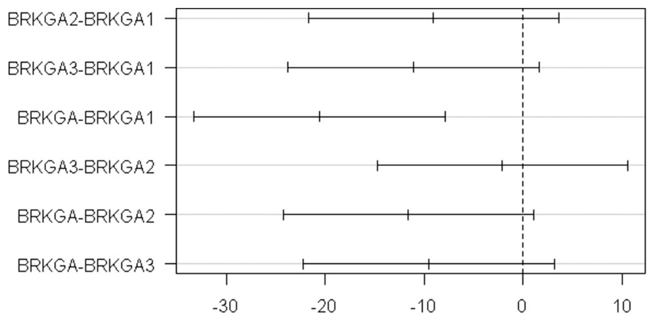

**Fig. 11.** Tukey's test between all BRKGA tested.

- JPO20 algorithm, proposed by Framinan and Perez-Gonzalez [10];
- SR2, the size-reduction SR-Q (algorithm under proposition), with the value of $Q$ equal to 2;
- DE, the recent differential evolution for order scheduling with setup times proposed by Prata et al. [21];
- BRKGA, the proposed metaheuristic with local search and parameter-free restart.

We added the four benchmarking approaches as a complementary reference (FP, NEH, OMDD and JPO20). In addition, we tested the EDD method. Since BRKGA uses EDD in the initial solution as a hybrid approach, it is possible to evaluate the improvement provided by BRKGA on the solution generated by EDD.

For the FP, OMDD and JPO20 the same parameters used by Framinan and Perez-Gonzalez [10] were considered in the tests with FP and JPO20 receiving OMDD as initial solution of algorithm, due to its capacity to generated competitive solution in admissible computational times [10]. For SR2, we too add OMDD with a warm-start of MILP. Thus, the algorithm does not fix the arcs of the OMDD solution for the initial solution to be valid. In DE and NEH we adapt the objective

**Table 6**
ARDI for all tested methods and instances sets.

| n | m | EDD | FP | NEH | OMDD | MILP | CP | JPO20 | SR2 | DE | BRKGA |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 100 | 5 | 97.44 | 52.53 | 43.67 | 9.88 | 4.73 | 2.12 | 5.06 | 5.71 | 11.75 | **0.31** |
|  | 10 | 98.86 | 47.91 | 39.88 | 12.56 | 5.42 | 4.92 | 5.64 | 8.66 | 14.73 | **0.30** |
| 150 | 5 | 96.04 | 50.41 | 47.55 | 10.06 | 12.71 | 2.75 | 6.30 | 6.80 | 19.44 | **0.25** |
|  | 10 | 98.03 | 50.76 | 43.90 | 11.02 | 15.05 | 5.90 | 9.16 | 8.08 | 26.99 | **0.05** |
| 200 | 5 | 92.27 | 46.48 | 48.89 | 10.08 | 23.80 | 3.25 | 8.61 | 7.11 | 24.88 | **0.11** |
|  | 10 | 92.86 | 48.08 | 41.49 | 9.76 | 38.67 | 6.28 | 9.28 | 8.03 | 32.61 | **0.12** |
| 300 | 5 | 45.21 | 24.03 | 20.97 | 4.98 | 94.97 | 2.04 | 4.86 | 4.34 | 14.33 | **0.03** |
|  | 10 | 49.51 | 25.16 | 19.43 | 5.51 | 97.69 | 4.25 | 5.49 | 4.96 | 20.61 | **0.00** |
| Total |  | 83.78 | 43.17 | 38.22 | 9.23 | 36.63 | 3.94 | 6.80 | 6.71 | 20.67 | **0.14** |

**Table 7**
ARDI for all tested methods and due dates factors.

| TF | RDD | EDD | FP | NEH | OMDD | MILP | CP | JPO20 | SR2 | DE | BRKGA |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.35 | 0.35 | 82.90 | 33.15 | 36.15 | 11.13 | 27.20 | 4.40 | 5.59 | 5.98 | 18.56 | **0.18** |
|  | 0.65 | 72.54 | 54.29 | 63.71 | 6.74 | 40.15 | 2.34 | 5.73 | 3.92 | 23.06 | **0.38** |
| 0.65 | 0.35 | 90.77 | 33.38 | 26.66 | 11.36 | 34.92 | 5.14 | 8.51 | 9.81 | 21.56 | **0.02** |
|  | 0.65 | 88.91 | 51.86 | 26.38 | 7.71 | 44.25 | 3.88 | 7.37 | 7.14 | 19.49 | **0.00** |
| Total |  | 83.78 | 43.17 | 38.22 | 9.23 | 36.63 | 3.94 | 6.80 | 6.71 | 20.67 | **0.14** |

function for order scheduling with missing operations and remove the filters in local search due there are no setup times in the proposed variant. In addition, we use the following parameters for DE: NP = $2\times n$, $G_y = 25$ and CR = 0.85 [21].

### 5.4. Results and discussion

Table 6 shows the average RDI (ARDI) of tested methods in each set of instances. The MILP and CP found feasible solutions for all the test instances. To analyze the performance of the tested methods with the factors controlling the range of due dates, Table 7 illustrates the results of the proposed methods concerning the $TF$ and $RDD$ factors.

Due the large values of $n$ 100 to 300 jobs, the BRKGA in all instances reaches the time limit before reaches the iteration limit. Analyzing Tables 6 and 7 the methods with the most competitive results (with ARDI less than 10) are OMDD, CP, JPO20, SR2 and BRKGA. The results prove that the pure MILP model is more effective for smaller instances, such as 100 orders and 5 or 10 machines. On the other hand, the SR2 method is more effective for intermediate instances, between 150 and 200 orders with 5 or 10 machines. For larger instances, the use of the OMDD heuristic proved to be more effective when compared to MILP and SR2. For better visualization of the results, Fig. 12 illustrates a boxplot with the RDI of all methods at all instance sizes. The Figs. 13, 14, and 15 illustrate the RDI results in a boxplot grouped by due date factors $TF$ and $RDD$ and density of missing operations ($\Theta$), respectively. In all next charts just the best eight methods are reported for a better visualization of results.

Overall, BRKGA obtained the best results, followed by CP, SR2 and JPO20. Comparing the matheuristic method, CP and MILP and the tested metaheuristics, the proposed BRKGA has the lowest interquartile range, median, and outliers values. The methods EDD, FP, MILP and NEH got the worst results. BRKGA achieved a significant improvement on the initial EDD solution. Therefore, improving the efficient local search and the restart operator along with BRKGA iterations can substantially improve the solution of the priority rule. For a better comparison, Fig. 16 illustrates the ARDI of each of the tested methods, grouped by job size.

From Fig. 16, NEH has poor performance compared to other strategies up to instance size 200, and the NEH performance with instance size 300 has similar performance to other methods. The MILP method works well up to an instance size of 200. At large sizes, this method gave the worst results of all the methods. The BRKGA, SR2, and CP methods produced competitive results for most instance sizes, and

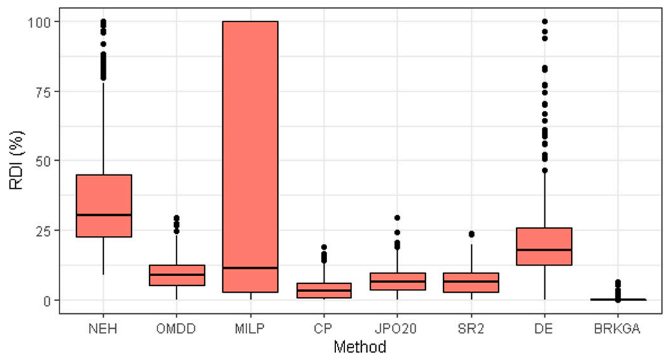

**Fig. 12.** RDI distribution for all tested methods in all sets of instances.

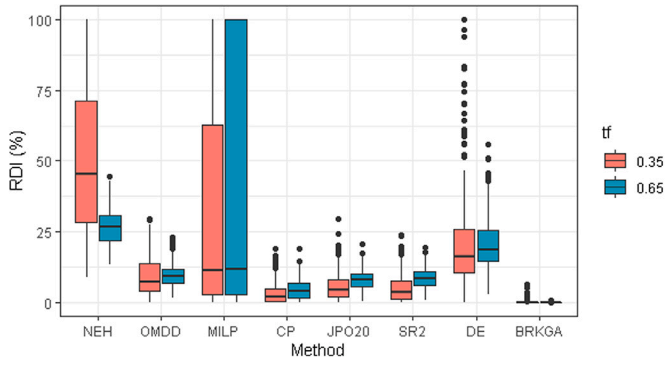

**Fig. 13.** RDI distribution for all tested methods in all sets of instances grouped by $TF$.

BRKGA produced better results. To verify performance differences between the approximated methods tested, Fig. 17 illustrates the ARDI for each of the approximated methods (matheuristics and metaheuristics).

From Fig. 17, DE has poor performance compared to other strategies in all instance sizes with the most variability of results. In addition, JPO20 has an ARDI of less than 10% in each instance size but is outperformed by BRKGA, which has competitive performance between matheuristics and metaheuristics methods with an ARDI of less than 1% in each instance size.

Figs. 16 and 17 show that BRKGA method outperforms most other algorithms in many instance sizes, with significant differences even

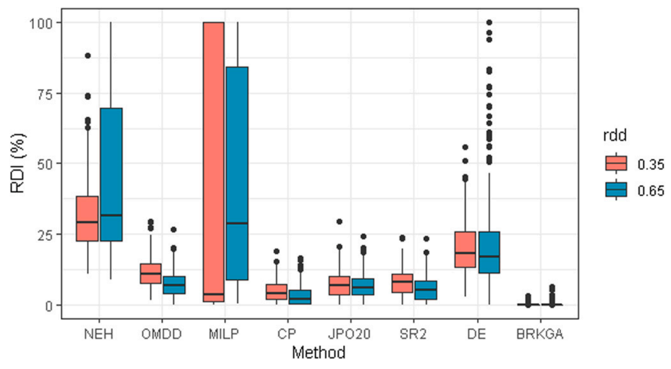

**Fig. 14.** RDI distribution for all tested methods in all sets of instances grouped by *RDD*.

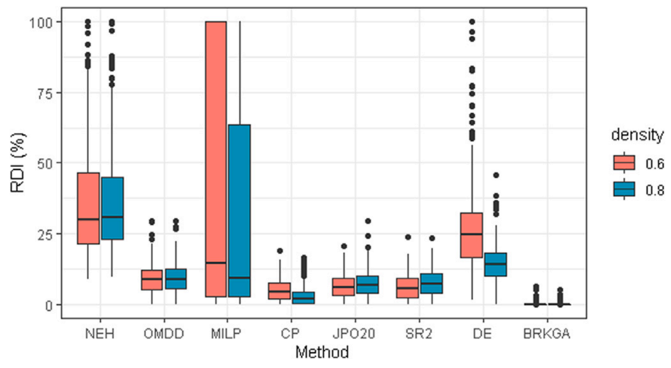

**Fig. 15.** RDI distribution for all tested methods in all sets of instances grouped by percentage of missing operations in the process times matrix ($\Theta$).

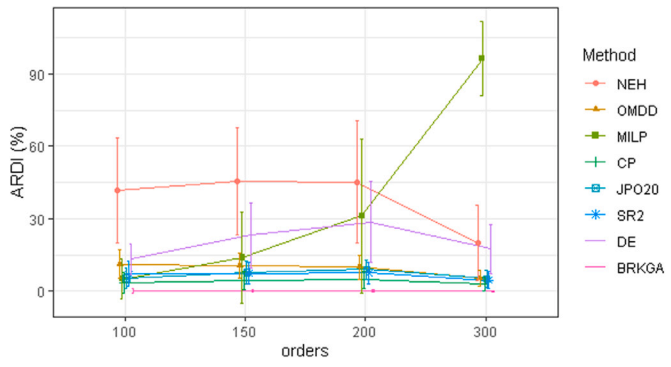

**Fig. 16.** ARDI and confidence interval ($\alpha$ = 0.05) for all tested methods in each job size.

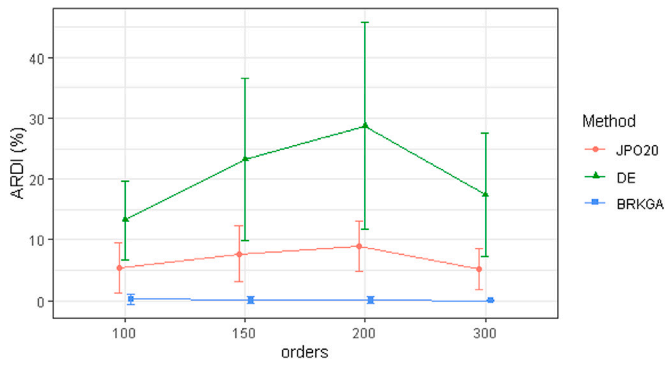

**Fig. 17.** ARDI and confidence interval ($\alpha$ = 0.05) for matheuristic and metaheuristic methods in each job size.

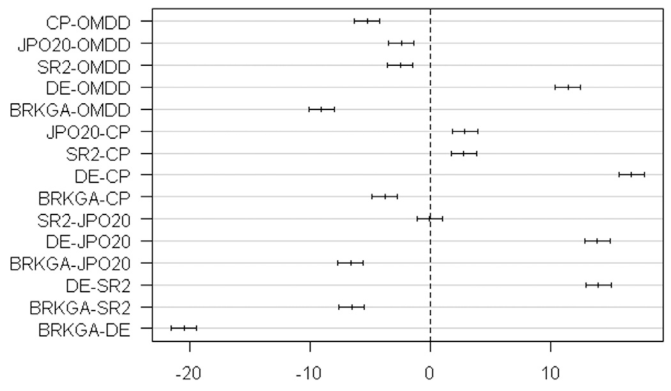

**Fig. 18.** Tukey's test between all tested methods.

within the approximate methods, primarily with average instance sizes of 200–300, and confidence intervals that do not intersect with most other methods. For analyzing the results of all instances in each method, Table 8 describes SR values for each method. The BRKGA got the best result in 84% of instance set.

To check if the differences between methods in the ARID performance indicator are statistically significant, we applied the analysis of variance (ANOVA). The *p*-value obtained is very close to zero, indicating that the results are significant. Finally, we applied Tukey's test in Fig. 18 to the tested data to check which ARID differences are significant between the benchmarks methods and the BRKGA algorithm. BRKGA obtained significantly better results when compared peer-to-peer with all other benchmarking methods, outperforming CP, MILP, SR and the most recent metaheuristic DE (the confidence intervals of each BRKGA comparison do not cross the zero line demonstrating significant results).

Therefore, analyzing the results present in Tables 6 and 7 and Figs. 12 and 16. BRKGA, CP and SR2 methods obtained better solutions than all other methods with SR2 performed slightly better on average than the JPO20 but the result is not significant in Tukey's test. The proposed CP model outperformed MILP, becoming the best exact model for the problem. BRKGA outperformed all tested benchmarking methods concerning solution quality. Therefore, BRKGA got an ARDI as lower as 0.15% and outperformed all other exact and approximation methods with significant solution quality in Tukey's test. The BRKGA is so far considered a competitive meta-heuristic for order scheduling with missing operations and total tardiness minimization, with a good trade-off between solution quality and admissible computational cost.

## 6. Final remarks and perspectives

In this paper, we investigate a new variant for the customer order scheduling problem with missing operations to minimize the total tardiness. We introduce an innovative size reduction algorithm with partitions for a scheduling problem with due dates. This approach does not take into account the processing times as the early approaches, but an efficient partitioning of the decision variables based on the due dates and a dispatch rule. In addition, we propose a new BRKGA metaheuristic with an efficient 2-opt local search and a parameter-free restart procedure.

Computational experiments with randomly generated test instances were carried out to evaluate the performance of the proposed solution procedures with the average relative deviation index as performance measure. In most tested problem instances, the BRKGA approach show the best performance, outperforming the JPO20 algorithm, the best algorithm in the literature adapted for the variant under study and DE, the most recent metaheuristic for a closely related variant.

Some advantages of BRKGA are an intensive local search phase to intensity quality search and a parameter-free restart procedure to

**Table 8**
Success rate for each method.

| Methods | EDD | FP | NEH | OMDD | MILP | CP | JPO20 | SR2 | DE | BRKGA |
|---|---|---|---|---|---|---|---|---|---|---|
| SR (%) | 0 | 0 | 0 | 0 | 4 | 11 | 0 | 4 | 0 | **84** |

reduce the number of parameters in calibration processes. However, BRKGA has disadvantages in considering just one population at a time, reducing the number of solutions that can be created at the same time [42]. Another disadvantage is the lack of hybridization with quality exact approaches, such as the CP model, for improving the search or the initial solution construction for BRKGA.

As a suggestion for future developments, we recommend the hybridization of constraint programming and metaheuristics to improve the solutions generated by the proposed methods. Future research could also consider other performance measures, such as the total completion time minimization.

## CRediT authorship contribution statement

**Levi Ribeiro de Abreu:** Conceptualization, Methodology, Investigation, Software, Formal analysis, Writing – original draft, Visualization, Writing – review & editing. **Bruno de Athayde Prata:** Conceptualization, Methodology, Validation, Writing – review & editing, Supervision. **Allan Costa Gomes:** Conceptualization, Methodology, Investigation, Software, Formal analysis, Writing – original draft. **Stéphanie Alencar Braga-Santos:** Conceptualization, Methodology, Investigation, Software, Formal analysis, Writing – original draft. **Marcelo Seido Nagano:** Conceptualization, Validation, Writing – review & editing, Supervision, Project administration.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

This work was supported by the Brazilian National Council for Scientific and Technological Development (Conselho Nacional de Desenvolvimento Científico e Tecnológico) [Nos. 309755/2021-2, 407151/2021-4, and 312585/2021-7]; the Brazilian Coordination for the Improvement of Higher Education Personnel (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) [No. 001]; the São Paulo Research Foundation (FAPESP) - through Grant no. #2021/11586-2.

## References

[1] J.M. Framinan, P. Perez-Gonzalez, V. Fernandez-Viagas, Deterministic assembly scheduling problems: A review and classification of concurrent-type scheduling models and solution procedures, European J. Oper. Res. 273 (2) (2019) 401–417.
[2] C.-T. Tseng, C.-J. Liao, T.-X. Liao, A note on two-stage hybrid flowshop scheduling with missing operations, Comput. Ind. Eng. 54 (3) (2008) 695–704.
[3] R. Leisten, M. Kolbe, A note on scheduling jobs with missing operations in permutation flow shops, Int. J. Prod. Res. 36 (9) (1998) 2627–2630.
[4] C. Rajendran, H. Ziegler, A performance analysis of dispatching rules and a heuristic in static flowshops with missing operations of jobs, European J. Oper. Res. 131 (3) (2001) 622–634.
[5] M. Dios, V. Fernandez-Viagas, J.M. Framinan, Efficient heuristics for the hybrid flow shop scheduling problem with missing operations, Comput. Ind. Eng. 115 (2018) 88–99.
[6] J. Sridhar, C. Rajendran, Scheduling in a cellular manufacturing system: a simulated annealing approach, Int. J. Prod. Res. 31 (12) (1993) 2927–2945.
[7] F. Zhao, X. He, L. Wang, A two-stage cooperative evolutionary algorithm with problem-specific knowledge for energy-efficient scheduling of no-wait flow-shop problem, IEEE Trans. Cybern. 51 (11) (2020) 5291–5303.
[8] F. Zhao, R. Ma, L. Wang, A self-learning discrete jaya algorithm for multiobjective energy-efficient distributed no-idle flow-shop scheduling problem in heterogeneous factory system, IEEE Trans. Cybern. 3086181 (2021) 1–12.
[9] I.S. Lee, Minimizing total tardiness for the order scheduling problem, Int. J. Prod. Econ. 144 (1) (2013) 128–134.
[10] J.M. Framinan, P. Perez-Gonzalez, Order scheduling with tardiness objective: Improved approximate solutions, European J. Oper. Res. 266 (3) (2018) 840–850.
[11] C.S. Sung, S.H. Yoon, Minimizing total weighted completion time at a pre-assembly stage composed of two feeding machines, Int. J. Prod. Econ. 54 (3) (1998) 247–255, URL: http://www.sciencedirect.com/science/article/pii/S0925527397001515.
[12] R. Ahmadi, U. Bagchi, T.A. Roemer, Coordinated scheduling of customer orders for quick response, Nav. Res. Logist. 52 (6) (2005) 493–512.
[13] J.Y.T. Leung, H. Li, M. Pinedo, Order scheduling in an environment with dedicated resources in parallel, J. Sched. 8 (5) (2005) 355–386.
[14] C.-C. Wu, T.-H. Yang, X. Zhang, C.-C. Kang, I.-H. Chung, W.-C. Lin, Using heuristic and iterative greedy algorithms for the total weighted completion time order scheduling with release times, Swarm Evol. Comput. 44 (2019) 913–926, URL: http://www.sciencedirect.com/science/article/pii/S2210650218301317.
[15] Z. Shi, L. Wang, P. Liu, L. Shi, Minimizing completion time for order scheduling: Formulation and heuristic algorithm, IEEE Trans. Autom. Sci. Eng. 14 (4) (2017) 1558–1569.
[16] J.M. Framinan, P. Perez-Gonzalez, New approximate algorithms for the customer order scheduling problem with total completion time objective, Comput. Oper. Res. 78 (2017) 181–192.
[17] V. Riahi, M.H. Newton, M. Polash, A. Sattar, Tailoring customer order scheduling search algorithms, Comput. Oper. Res. 108 (2019) 155–165, URL: http://www.sciencedirect.com/science/article/pii/S030505481930098X.
[18] J.-Y. Kung, J. Duan, J. Xu, I. Chung, S.-R. Cheng, C.-C. Wu, W.-C. Lin, et al., Metaheuristics for order scheduling problem with unequal ready times, Discrete Dyn. Nat. Soc. 2018 (2018) 1–13.
[19] W.-C. Lin, J. Xu, D. Bai, I.-H. Chung, S.-C. Liu, C.-C. Wu, Artificial bee colony algorithms for the order scheduling with release dates, Soft Comput. 23 (18) (2019) 8677–8688.
[20] B.A. Prata, C.D. Rodrigues, J.M. Framinan, Customer order scheduling problem to minimize makespan with sequence-dependent setup times, Comput. Ind. Eng. 151 (2021) 106962.
[21] B.d.A. Prata, C.D. Rodrigues, J.M. Framinan, A differential evolution algorithm for the customer order scheduling problem with sequence-dependent setup times, Expert Syst. Appl. 189 (2022) 116097.
[22] M.P. Antonioli, C.D. Rodrigues, B.d.A. Prata, Minimizing total tardiness for the order scheduling problem with sequence-dependent setup times using hybrid matheuristics, Int. J. Ind. Eng. Comput. 13 (2) (2022) 1–24.
[23] F. Rossi, P. Van Beek, T. Walsh, Handbook of Constraint Programming, Elsevier, 2006.
[24] M. Pinedo, Scheduling: Theory, Algorithms, and Systems, Springer, New York, 2016.
[25] P. Laborie, D. Godard, Self-adapting large neighborhood search: Application to single-mode scheduling problems, in: Proceedings MISTA-07, Paris, vol. 8, Citeseer, 2007.
[26] M.F. Zarandi, H. Khorshidian, M.A. Shirazi, A constraint programming model for the scheduling of JIT cross-docking systems with preemption, J. Intell. Manuf. 27 (2) (2016) 297–313.
[27] R. Gedik, D. Kalathia, G. Egilmez, E. Kirac, A constraint programming approach for solving unrelated parallel machine scheduling problem, Comput. Ind. Eng. 121 (2018) 139–149.
[28] P. Yunusoglu, S. Topaloglu Yildiz, Constraint programming approach for multi-resource-constrained unrelated parallel machine scheduling problem with sequence-dependent setup times, Int. J. Prod. Res. (2021) 1–18.
[29] H. Öztop, M.F. Tasgetiren, L. Kandiller, Q.-K. Pan, Metaheuristics with restart and learning mechanisms for the no-idle flowshop scheduling problem with makespan criterion, Comput. Oper. Res. 138 (2022) 105616.
[30] J. Kelbel, Z. Hanzálek, Solving production scheduling with earliness tardiness penalties by constraint programming, J. Intell. Manuf. 22 (4) (2011) 553–562.
[31] L. Meng, C. Zhang, Y. Ren, B. Zhang, C. Lv, Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem, Comput. Ind. Eng. 142 (2020) 106347.
[32] L.R. Abreu, K.A.G. Araújo, B.d.A. Prata, M.S. Nagano, J.V. Moccellin, A new variable neighbourhood search with a constraint programming search strategy for the open shop scheduling problem with operation repetitions, Eng. Optim. (2021) 1–20.
[33] L.R. Abreu, M.S. Nagano, A new hybridization of adaptive large neighborhood search with constraint programming for open shop scheduling with sequence-dependent setup times, Comput. Ind. Eng. 168 (2022) 108128.
[34] B. Naderi, S.F. Ghomi, M. Aminnayeri, M. Zandieh, Modeling and scheduling open shops with sequence-dependent setup times to minimize total completion time, Int. J. Adv. Manuf. Technol. 53 (5–8) (2011) 751–760.

[35] S. Braga-Santos, G. Barroso, B. Prata, A size-reduction algorithm for the order scheduling problem with total tardiness minimization, J. Project Manage. 7 (3) (2022) 167–176.
[36] L. Fanjul-Peyro, R. Ruiz, Size-reduction heuristics for the unrelated parallel machines scheduling problem, Comput. Oper. Res. 38 (1) (2011) 301–309, Project Management and Scheduling.
[37] F.E. Achamrah, F. Riane, C. Di Martinelly, S. Limbourg, A matheuristic for solving inventory sharing problems, Comput. Oper. Res. 138 (2022) 105605.
[38] B. de Athayde Prata, V. Fernandez-Viagas, J.M. Framinan, C.D. Rodrigues, Matheuristics for the flowshop scheduling problem with controllable processing times and limited resource consumption to minimize total tardiness, Comput. Oper. Res. (2022) 105880.
[39] J.F. Gonçalves, M.G. Resende, Biased random-key genetic algorithms for combinatorial optimization, J. Heuristics 17 (5) (2011) 487–525.
[40] G.-G. Wang, D. Gao, W. Pedrycz, Solving multi-objective fuzzy job-shop scheduling problem by a hybrid adaptive differential evolution algorithm, IEEE Trans. Ind. Inf. (2022).
[41] S. Zhou, L. Xing, X. Zheng, N. Du, L. Wang, Q. Zhang, A self-adaptive differential evolution algorithm for scheduling a single batch-processing machine with arbitrary job sizes and release times, IEEE Trans. Cybern. 51 (3) (2019) 1430–1442.
[42] C.E. Andrade, R.F. Toso, J.F. Gonçalves, M.G. Resende, The multi-parent biased random-key genetic algorithm with implicit path-relinking and its real-world applications, European J. Oper. Res. 289 (1) (2021) 17–30.
[43] M. Kong, X. Liu, J. Pei, H. Cheng, P.M. Pardalos, A BRKGA-DE algorithm for parallel-batching scheduling with deterioration and learning effects on parallel machines under preventive maintenance consideration, Ann. Math. Artif. Intell. 88 (1) (2020) 237–267.
[44] J. Rocholl, L. Mönch, Decomposition heuristics for parallel-machine multiple orders per job scheduling problems with a common due date, J. Oper. Res. Soc. 72 (8) (2021) 1737–1753.
[45] L. Abreu, B. Prata, A hybrid genetic algorithm for solving the unrelated parallel machine scheduling problem with sequence dependent setup times, IEEE Lat. Am. Trans. 16 (6) (2018) 1715–1722.
[46] L.R. de Abreu, B. de Athayde Prata, A genetic algorithm with neighborhood search procedures for unrelated parallel machine scheduling problem with sequence-dependent setup times, J. Model. Manage. (2020).
[47] L.S. Pessoa, C.E. Andrade, Heuristics for a flowshop scheduling problem with stepwise job objective function, European J. Oper. Res. 266 (3) (2018) 950–962.
[48] C.E. Andrade, T. Silva, L.S. Pessoa, Minimizing flowtime in a flowshop scheduling problem with a biased random-key genetic algorithm, Expert Syst. Appl. 128 (2019) 67–80.
[49] I. Maciel, B. Prata, M. Nagano, L. Abreu, A hybrid genetic algorithm for the hybrid flow shop scheduling problem with machine blocking and sequence-dependent setup times, J. Project Manage. 7 (4) (2022) 191–225.
[50] M.T. Pereira, M.S. Nagano, Hybrid metaheuristics for the integrated and detailed scheduling of production and delivery operations in no-wait flow shop systems, Comput. Ind. Eng. (2022) 108255.
[51] N.d.C.L. Beirão, Sistema de apoio à decisão para sequenciamento de operações em ambientes Job Shop, Faculdade de Engenharia do Porto, 1997.
[52] J.F. Gonçalves, J.J. de Magalhães Mendes, M.G. Resende, A hybrid genetic algorithm for the job shop scheduling problem, European J. Oper. Res. 167 (1) (2005) 77–95.
[53] S.M. Homayouni, D.B. Fontes, J.F. Gonçalves, A multistart biased random key genetic algorithm for the flexible job shop scheduling problem with transportation, Int. Trans. Oper. Res. 00 (2020) 1–29.
[54] L.R. Abreu, J.O. Cunha, B.A. Prata, J.M. Framinan, A genetic algorithm for scheduling open shops with sequence-dependent setup times, Comput. Oper. Res. 113 (2020) 104793.
[55] L.R. Abreu, R.F. Tavares-Neto, M.S. Nagano, A new efficient biased random key genetic algorithm for open shop scheduling with routing by capacitated single vehicle and makespan minimization, Eng. Appl. Artif. Intell. 104 (2021) 104373.
[56] L.R. Abreu, B.A. Prata, J.M. Framinan, M.S. Nagano, New efficient heuristics for scheduling open shops with makespan minimization, Comput. Oper. Res. 142 (2022) 105744.
[57] M. López-Ibáñez, J. Dubois-Lacoste, L.P. Cáceres, M. Birattari, T. Stützle, The irace package: Iterated racing for automatic algorithm configuration, Oper. Res. Perspect. 3 (2016) 43–58.
[58] J.V. Moccellin, M.S. Nagano, A.R. Pitombeira Neto, B. de Athayde Prata, Heuristic algorithms for scheduling hybrid flow shops with machine blocking and setup times, J. Braz. Soc. Mech. Sci. Eng. 40 (2) (2018) 1–11.
[59] A.R. Pitombeira-Neto, B.d.A. Prata, A matheuristic algorithm for the one-dimensional cutting stock and scheduling problem with heterogeneous orders, Top 28 (1) (2020) 178–192.
[60] F.S.d. Almeida, M.S. Nagano, Heuristics to optimize total completion time subject to makespan in no-wait flow shops with sequence-dependent setup times, J. Oper. Res. Soc. (2022) 1–12.
[61] M.F. Rego, J.C.E. Pinto, L.P. Cota, M.J. Souza, A mathematical formulation and an NSGA-II algorithm for minimizing the makespan and energy cost under time-of-use electricity price in an unrelated parallel machine scheduling, PeerJ Comput. Sci. 8 (2022) e844.
[62] D.C. Montgomery, Design and Analysis of Experiments, John Wiley & Sons, 2017.