TOP (2021) 29:524–546
https://doi.org/10.1007/s11750-020-00574-x

ORIGINAL PAPER

Heuristic methods for the single-machine scheduling problem with periodical resource constraints

Bruno de Athayde Prata[1] · Levi Ribeiro de Abreu[1] · José Ytalo Ferreira Lima[1]

Received: 15 July 2019 / Accepted: 21 June 2020 / Published online: 3 July 2020

Abstract
In the last years, researchers have been paying special attention to scheduling problems with scarce resource consumption and periodic maintenance activities with a view to the adoption of more realistic assumptions. This paper aims at presenting heuristics to a single-machine scheduling environment with periodical resource constraints. In this new variant of the single-machine scheduling problem, in each production period, there are resource consumption constraints. To the best of our knowledge, the proposed variant has not been addressed in the revised literature. An integer linear programming model is presented based on a bin packing formulation taking two packing constraints into consideration. The objective function adopted is makespan minimization, and relative deviation is used as a performance criterion. Since the problem under study is NP-hard, heuristic algorithms are proposed to obtain high-quality solutions in acceptable computational times. Eighteen constructive heuristics, two local search heuristics, and a hybrid matheuristics, based on the size reduction and simulated annealing algorithms, have been presented. The extensive computational experience carried out shows that the proposed local search algorithms, as well as the proposed matheuristics, are promising tools to solve large-sized instances.

Keywords Production sequencing · Scarce resources · Makespan · Size reduction

Mathematics Subject Classification 90B35 · 90C59

✉ Bruno de Athayde Prata
baprata@ufc.br

Levi Ribeiro de Abreu
leviribeiro@alu.ufc.br

José Ytalo Ferreira Lima
ytalolimah@gmail.com

[1] Department of Industrial Engineering, Federal University of Ceará, Campus do Pici, Fortaleza, Brazil

1 Introduction

In the manufacturing environment, the production processes are limited not only by the capacity of the machines but also by additional scarce resources. Several practical examples of this assumption in the electronic industry, as well as in the operations management, are presented by Ventura and Kim (2003).

Another example is observed in the construction industry. The production process of precast beams is constrained by the capacity of forms (molds) as well as materials and labor. In the production of large-sized precast elements, the occurrence of a single track of large dimensions is usually verified, characterizing a single-machine environment. Another issue is resource consumption, which is characterized by each production period (Prata et al. 2015).

The consideration of resource consumption in the parallel machine production environment has been studied in the last few years, given its theoretical and practical importance.

Ventura and Kim (2003) introduce a parallel machine environment with earliness–tardiness penalties and resource constraints. Edis and Ozkarahan (2012) present a real-world problem in an injection molding of an electrical appliance plant. This production environment is a resource-constrained parallel machine with eligibility restrictions. Ji et al. (2013) and Yeh et al. (2015) address the uniform parallel machine scheduling with resource consumption constraints. On the other hand, Zheng and Wang (2016), as well as Zheng and Wang (2018), address the resource-constrained unrelated parallel machines environment. Afzalirad and Rezaeian (2016) study an unrelated parallel machine scheduling problem with resource constraints, sequence-dependent setup times, different release dates, machine eligibility, and precedence constraints. Afzalirad and Shafipour (2018) address a resource-constrained unrelated parallel machine scheduling problem with eligibility constraints. Fanjul-Peyro et al. (2017) introduce a parallel machine scheduling problem in which the job processing requires a given amount of a scarce resource. The resources are limited and fixed over the production horizon. Villa et al. (2018) address the unrelated parallel machine scheduling problem with one scarce additional resource.

Although the resource-constrained parallel machine environment offers several contributions, studies addressing single-machine problems with resource consumption are still quite limited. Wang and Wang (2012) introduce a single-machine scheduling problem where the processing time of a given job is calculated using a resource consumption function. The objective function is the minimization of resource consumption. Wang and Wang (2013) approach a single-machine variant with resource-dependent processing times and deteriorating jobs. The aggregated objective function considers makespan, total completion time, total absolute differences in completion times and total resource cost, and a cost function. Zhu et al. (2013) present a single-machine scheduling problem in which the processing time of the current job is determined, among other factors, by a resource allocation function. Since the production times are usually dependent on the available resources, this is a realistic assumption. Wu and Cheng (2016) present a single-machine problem with resource constraints. The proposed variant arises in cloud computing applications. For each job, the processing time is a function of the assigned resources for its production. This assumption leads to a nonlinear combinatorial optimization problem. Karhi and Shabtay (2018) address a single-machine scheduling environment in a flexible framework, in which the job processing times and due dates are decision variables.

Recent studies have presented the consideration of periodic maintenance activities for single-machine scheduling. Ji et al. (2007) consider a single-machine problem with multiple maintenance periods in which each maintenance operation is performed after a periodic time interval. The performance measure is the makespan minimization and the well-known LPT priority rule is applied as a solution method, with a worst-case ratio equal to 2. Low et al. (2010a) present a particle swarm optimization (PSO) metaheuristics for the single-machine scheduling with periodical maintenance. The objective function is the makespan minimization. The above-mentioned authors have shown that the problem is NP-hard using a conversion in the 3-partition problem. Hsu et al. (2010) present an integer linear programming modeling for the single-machine scheduling problem with periodic maintenance activities to minimize makespan. Two heuristics based on the well-known best-fit algorithm for cutting and packing problems are proposed. Perez-Gonzalez and Framinan (2018) address the single-machine scheduling problem with cyclical machine availability periods for the makespan minimization. These authors performed extensive computational experimentation, comparing several constructive algorithms with the proposed integer programming model.

Ángel Bello et al. (2011) introduce the single-machine scheduling problem with programmed preventive maintenance and sequence-dependent setup times. A mixed-integer linear programming model, as well as a GRASP metaheuristics, are presented to tackle the problem. Pacheco et al. (2013) present a multi-start tabu search that outperforms the GRASP proposed by Ángel Bello et al. (2011).

Although the single-machine scheduling problem has been widely studied in the last decades, works addressing this environment, taking the resource consumption constraints per period into consideration, are rather limited. However, this paper introduces a variant of a single machine, taking resource consumption with a periodical availability into account. In the problem under study, there is a single-machine environment with multiple production periods in which each period presents a duration of T time units. Furthermore, in each production period, there is also a resource constraint R that restricts resource consumption. This resource constraint is fixed by each production period so that the resource is fully available in the next one. This situation arises in many real-world problems related to the molding of products, such as plastic or precast concrete production.

In this paper, heuristics for the single-machine scheduling problem with periodical resource constraints (SMPRC) are proposed. There are four main contributions in this study. First, a mixed-integer programming formulation for the SMPRC is presented. According to the literature review, this variant has not been modeled yet, despite its theoretical and practical importance. From a practical point of view, the proposed variant is important because there are several production problems in which a single machine with periodical constraints appears, for instance, in the production of molded items. From a theoretical point of view, the proposed variant combines a scheduling environment with a bin packing problem, which is similar to the bin packing presented by Fanjul-Peyro et al. (2017), in which there are two types of packing. Secondly, some problem properties, such as the NP-hardness proof and a lower bound, are presented. Third, a set of eighteen constructive heuristics for the problem under study, as well as two local search heuristics, are proposed. Thereafter, a matheuristics is proposed to find near-optimal solutions in acceptable computational times.

This paper is organized as follows: in Sect. 2, the mixed-integer linear programming (MILP) model is described; in Sect. 3, a hybrid size reduction and simulated annealing algorithm is proposed; in Sect. 4, some outcomes from computational experiments are discussed; finally, in Sect. 5, some conclusions and suggestions for future works are presented.

2 Problem statement and MILP model

2.1 Problem definition

Let a single machine that produces n jobs in which each job has an associated processing time as well as resource consumption. There is a resource consumption constraint for each production period; thus, the production planning is divided into periods, such as a workday. In the variant under study, the following assumptions are considered: (1) each job is processed by a single machine; (2) each machine can process a single job at the same time; (3) preemption of jobs is forbidden; (4) each job presents associated processing times and a fixed amount of resources per period; and (5) each production period presents a time constraint as well as a resource consumption constraint. The objective function is the makespan minimization, taking the processing times and the resource consumptions for each job into account. The problem is similar to a bin packing problem with two characteristics: processing times and resource consumption. In each production period, the jobs must be processed considering both time and resource constraints. A period could be viewed as a bin with two capacities: duration T, as well as an amount of available resources R.

It is assumed that p is the processing time vector, r is the resource consumption vector, T is the duration of the production periods, and R is the maximal amount of resources available per period. It is possible to observe that the periods have the same maximum amount of resources and these resources are not renewable. In addition, the periods organized a posteriori in non-increasing order not to unnecessarily increase the makespan.

An illustrative example with five jobs, $p = \{2, 1, 5, 4, 3\}$, $r = \{3, 3, 4, 1, 1\}$, $T = 5$, and $R = 4$ is considered. A feasible solution for this instance is the sequence $\Pi = \{1, 5, 2, 4, 3\}$, with three periods with a makespan of 15 time units, as illustrated in Fig. 1.

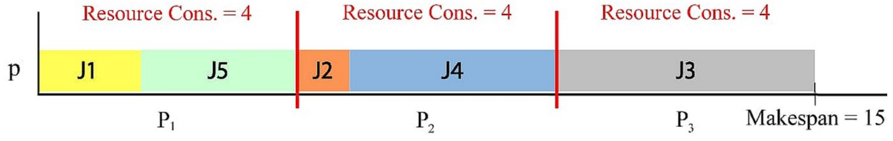

Fig. 1 Gantt chart for the presented solution

2.2 Proposed mixed-integer programming model

Let a set of n jobs with a resource consumption to be processed in a machine subjected to two constraints: a period constraint and a resource constraint. In the problem under study, the jobs in the single machine are scheduled, minimizing the number of required periods. Hereafter, the notation used for the problem modeling is presented.

Indices and sets

i: index for periods $\{1, 2,..., t\}$.
j: index for jobs $\{1, 2,..., n\}$.

Parameters

p_j: processing time of job j.
r_j: required resource amount for job j.
T: Maximum duration of the periods.
R: Maximum amount of resource for each period.
M: a large positive number.

Decision variables

$$x_{ij} = \begin{cases} 1, & \text{if job } j \text{ is processed in period } i \\ 0, & \text{otherwise} \end{cases}$$

$$y_i = \begin{cases} 1, & \text{if period } i \text{ is used in a given solution} \\ 0, & \text{otherwise} \end{cases}$$

$$w_i = \begin{cases} 1, & \text{if it is the period with the longest idle time (maximum slack occurs in period } i) \\ 0, & \text{otherwise} \end{cases}$$

z: computes the longest idle time (maximum slack) in the last period.

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

minimize

$$T \cdot \sum_{i=1}^{t} y_i - z \tag{1}$$

subject to:

$$\sum_{i=1}^{t} x_{ij} = 1, \quad \forall j \tag{2}$$

$$\sum_{j=1}^{n} p_j x_{ij} \le T, \quad \forall i \tag{3}$$

$$\sum_{j=1}^{n} r_j x_{ij} \le R, \quad \forall i \tag{4}$$

$$x_{ij} \le y_i \quad \forall i,j \tag{5}$$

$$\sum_{i=1}^{t} w_i = 1 \tag{6}$$

$$w_i \le y_i \quad \forall i \tag{7}$$

$$z \le M(1 - w_i) + Ty_i - \sum_{j=1}^{n} p_j x_{ij}, \quad \forall i \tag{8}$$

$$y_i \ge 0 \quad \forall i \tag{9}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i,j \tag{10}$$

$$w_i \in \{0, 1\} \quad \forall i \tag{11}$$

$$z \ge 0. \tag{12}$$

Objective function (1) is the makespan minimization. Constraint set (2) ensures that a job is processed only in period i. Constraint set (3) enforces the time constraint of each period. Constraint set (4) determines that the jobs produced in a given production period do not exceed the period's available resource. Constraint set (5) determines that the jobs are produced only in the selected periods. Constraint (6) states that a given period provides the maximum slack. Constraint set (7) states that, if a period is not used, it does not provide the maximum slack. Constraint set (8) used periods. Finally, constraint sets (9), (10), (11) and (12) determine domain of the decision variables. Taking constraint set (6) into consideration, it is possible to relax the integrality of the decision variables y_i. Regarding constraint set (8), it is valid to emphasize that only one period will be $w_i = 1$, so the rest will always be $z \le M$. For that period $w_i = 1$ (the last one), z is the difference between the total fixed time of a period (T) and the sum of the processing times of the jobs of that period. The proposed model presents $t(n + 1)$ binary decision variables, $t + 1$ continuous decision variables, and $n + 4t + nt + 1$ integer linear constraints.

2.3 Problem properties

In this subsection, three properties for the SPMRC are presented. First, the NP-hardness of the proposed problem. Second, a lower bound for the SPMRC. Finally, an upper bound for the number of periods.

Theorem 1 *The SPMRC is NP-hard.*

Proof Taking constraint sets (3) or (4) into consideration , if $p_j = 0\ \forall j$ or if $r_j = 0\ \forall j$, the problem is a particular case of the bin packing problem, which is NP-hard (Martello 1990). □

Proposition 1 *A lower bound for SPMRC is given by*

$$LB = \left\lceil \max\left\{ \frac{\sum_{j=1}^{n} p_j}{T}, \frac{\sum_{j=1}^{n} r_j}{R} \right\} \right\rceil. \tag{13}$$

Proof If there is no loss in the productive capacity, the minimal number of periods is given by the greatest whole number among the following ratios: summation of processing times divided by maximum duration of the planning periods and summation of required resources divided by the maximum amount of resource for each period. □

To sequence problems with multiple periods, a key issue for the model behavior is the determination of an estimate for the number of planning periods (Pitombeira-Neto and Prata 2019). On the one hand, the greater the planning horizon, the more variables there are in the model. It is important to highlight that if there is an excessive amount of decision variables, they will not be a part of the global optimal solution and the performance of the model resolution solver will decrease. On the other hand, a small number of time periods will lead to model unfeasibility.

As an upper bound for the number of periods, the concept of the well-known next-fit (NF) algorithm (Scheithauer 2017) was adopted. Let NF_P be the number of periods taking only the processing times into account and NF_R, the number of periods considering only resource consumption, an upper bound for the number of periods is given by

$$\text{UB} = \max\{\text{NF}_T, \text{NF}_R\}. \tag{14}$$

3 Proposed algorithms

3.1 Constructive heuristics

Constructive heuristics present great practical importance since they are able to provide good solutions within fast (usually insignificant) computation times. Considering the single-machine environment, the application of constructive heuristics which exploit specific characteristics of the problem to improve its performance plays a key role (Birgin and Ronconi 2012). The constructive algorithms proposed by Perez-Gonzalez and Framinan (2018) have been extended, for the single-machine scheduling problem with cyclical machine availability periods and makespan minimization, in the problem under study. One can observe that resource consumption is not treated in Perez-Gonzalez and Framinan (2018).

All the developed heuristics present three construction phases. Phase 1 consists of the aggregation of criteria for each job. In this study, two criteria were considered: processing times and resource consumption for each job. Three possibilities for this aggregation were considered, with the measures: the sum, the average, and the maximal value of the two criteria. A sequence s with the results of the aggregation of values of two criteria was created with one measure, where s_j is the result of the aggregation of job j.

Phase 2 of the constructive heuristics sorts the vector with the criteria aggregation for each job, following some sorting algorithm proposed by Perez-Gonzalez and Framinan (2018). In the problem under study, two criteria were considered for each job: the resource consumed for its production and the processing time.

For the execution of the sorting algorithm, a vector s was created in which s_j is the result of the criteria aggregation of resource consumption and processing time for job j, following the measure adopted in Phase 1 of the constructive heuristics. Next, vector s is organized in a non-decrescent way taking the aggregate value of the considered criteria into account, as presented by Perez-Gonzalez and Framinan (2018).

Let a given aggregate criterion for each job be s_j, generated in phase 1. The sequence of aggregate criteria are organized in a non-decreasing order without loss of generality, $s_j \leq s_{j+1}, j = 1..., n-1$. The methods used for sequencing the jobs are described as follows.

- Decreasing (D), sorting the jobs in a non-crescent order, as the well-known LPT dispatch rule (Ji et al. 2007; Yu et al. 2014): $s_n, \dots, s_1$;
- A-Sharp (A), sorting the jobs inserting the largest job in the middle of the sequence (Low et al. 2010a, b): $s_2, s_4, \dots, s_{n-1}, s_n, s_{n-2}, \dots, s_3, s_1$;
- Inserting Low and High criteria (HILO), sorting the jobs always inserting the largest and the smallest jobs in alternation (Hsu et al. 2010): $s_n, s_1, s_{n-1}, s_2, s_{n-2}, s_3, \dots$.

Phase 3 consists of the job allocation previously sequenced in the machine using criteria related to the bin packing problem. However, differently from the classical algorithms which consider a single criterion, in our approach, the resource consumption and the processing times are concomitantly considered. The bin packing policies are described as follows.

- First fit (FF): always insert a job that fits in a given production period, taking processing times as well as resource consumption into consideration. If a job cannot be allocated in the existent periods, a new period is created. The algorithm is finished when all the jobs are scheduled.
- Best fit (BF): always insert a job that results in the smaller accumulated slack, concomitantly considering the processing times and resource consumption. If a job cannot be allocated in the existent periods, a new period is created. The algorithm is finished when all the jobs are scheduled.

Figure 2 illustrates the considered bin packing policies, as well as an illustrative example with the allocation of the jobs shown in a Gantt chart. The first method, denoted as None, solely consists in the allocation of the jobs according to the first-in-first-out rule. FF policy allocates the job whenever practicable, and the BF policy is committed to allocating the jobs leading to a minor slack in each planning period. Table 1 describes the eighteen proposed constructive heuristics for the problem under study.

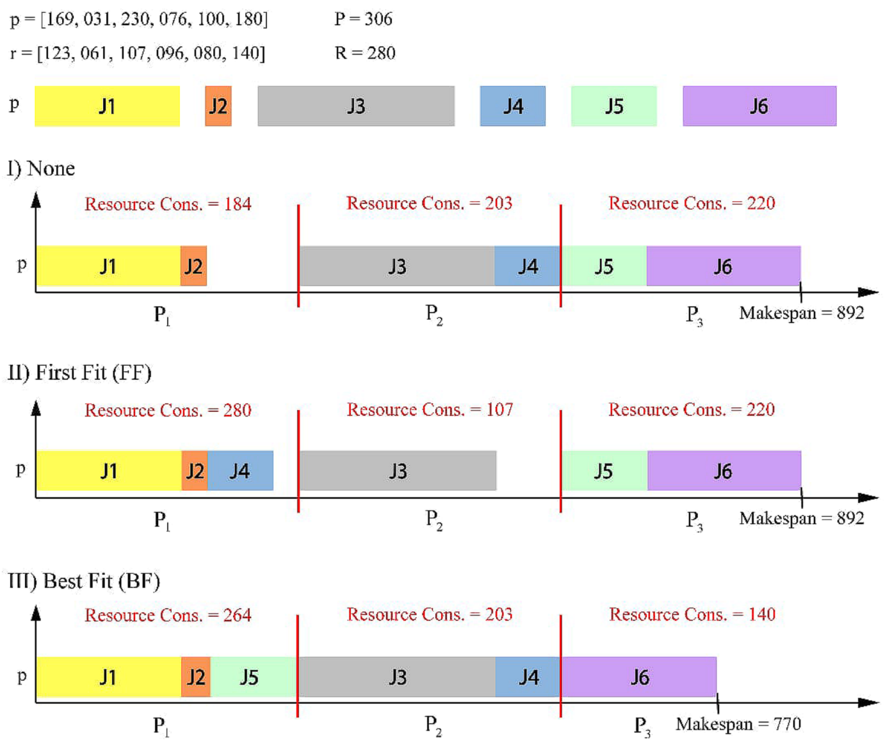

Fig. 2 Bin packing policies for an illustrative example

Table 1 Proposed constructive heuristics

Phase 1	Phase 2	Phase 3	Description	Notation
$p_i + r_i$	LPT	FF	Sum first fit decreasing	SFFD
$(p_i + r_i)/2$	LPT	FF	Avg first fit decreasing	AFFD
$\max(p_i, r_i)$	LPT	FF	Max first fit decreasing	MFFD
$p_i + r_i$	A-Sharp	FF	Sum first fit A-Sharp	SFFA
$(p_i + r_i)/2$	A-Sharp	FF	Avg first fit A-Sharp	AFFA
$\max(p_i, r_i)$	A-Sharp	FF	Max first fit A-Sharp	MFFA
$p_i + r_i$	HILO	FF	Sum first fit HILO	SFFHILO
$(p_i + r_i)/2$	HILO	FF	Avg first fit HILO	AFFHILO
$\max(p_i, r_i)$	HILO	FF	Max first fit HILO	MFFHILO
$p_i + r_i$	LPT	BF	Sum best fit decreasing	SBFD
$(p_i + r_i)/2$	LPT	BF	Avg best fit decreasing	ABFD
$\max(p_i, r_i)$	LPT	BF	Max best fit decreasing	MBFD
$p_i + r_i$	A-Sharp	BF	Sum best fit A-Sharp	SBFA
$(p_i + r_i)/2$	A-Sharp	BF	Avg best fit A-Sharp	ABFA
$\max(p_i, r_i)$	A-Sharp	BF	Max best fit A-Sharp	MBFA
$p_i + r_i$	HILO	BF	Sum best fit HILO	SBFHILO
$(p_i + r_i)/2$	HILO	BF	Avg best fit HILO	ABFHILO
$\max(p_i, r_i)$	HILO	BF	Max best fit HILO	MBFHILO

3.2 Proposed constructive heuristics with local search

For the single-machine and other scheduling problems, usually constructive heuristics lead to promising areas of search space and the generated solutions can be improved by local search procedures (Valente 2007). In this paper, two constructive heuristics followed by local search algorithms are proposed.

The proposed local search procedure is based on a neighborhood structure of a random insertion movement of a job in a solution sequence. Each job in the solution sequence is inserted in another random position. Figure 3 illustrates an example of a random insertion with 10 jobs in a given sequence.

If a new solution obtained after a random insertion is better than the best solution which was previously found, the current solution is replaced, and the remaining solutions of the previous neighborhood are not visited since the search restarts. If all solutions in a given neighborhood are evaluated, and there is no improvement, the search is finished, and the best solution is returned.

The local search procedure is described in detail by Framinan et al. (2014). The steps of the proposed local search heuristics are described as follows. The proposed heuristics are local search with first fit bin packing policy (LSFF) and local search with best fit bin packing policy (LSBF).

Step 1 : calculate the aggregate value of criteria (resource consumption and process time) using the max measure for each job;

Step 2 : sort the jobs according to the LPT rule applying this initial solution in the sequence of the aggregate value of criteria;

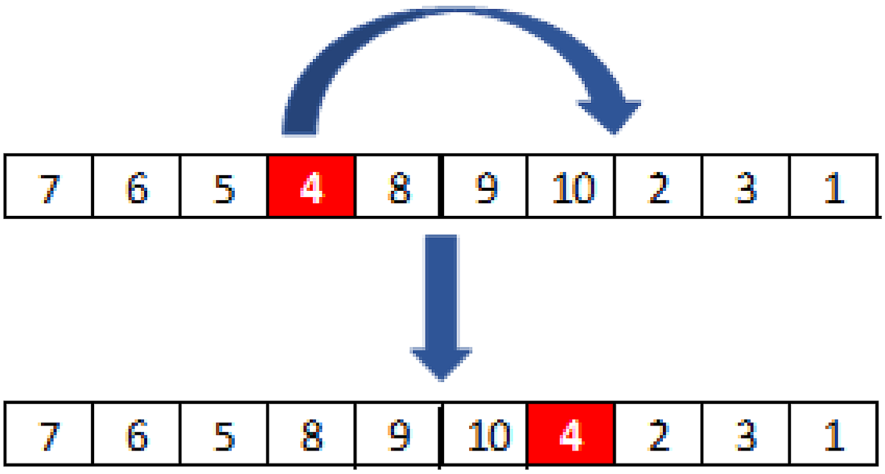

Fig. 3 Example of the random insertion procedure

Step 3 : utilize FF or BF bin packing policies in the job sequence;
Step 4 : apply random insertion local search with the first improvement procedure.

Figure 4 presents the flowchart of constructive heuristics with local search for FF or BF bin packing polices. The three first processes describe steps 1–3, and the next processes describe the random insertion local search with the first improvement procedure.

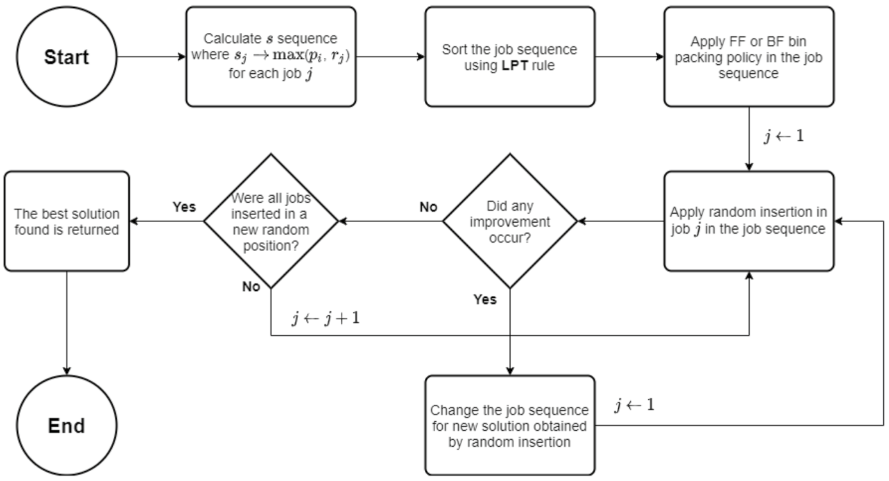
Start
Calculate s sequence where $s_j \to \max(p_i, r_j)$ for each job j
Sort the job sequence using **LPT** rule
Apply FF or BF bin packing policy in the job sequence
$j \leftarrow 1$
Apply random insertion in job j in the job sequence
Did any improvement occur?
No
Were all jobs inserted in a new random position?
Yes
The best solution found is returned
No
$j \leftarrow j+1$
Yes
Change the job sequence for new solution obtained by random insertion
$j \leftarrow 1$
End

Fig. 4 Flowchart of LSFF and LSBF constructive heuristics

3.3 Proposed matheuristics

In the last years, mathematical programming solvers have presented a substantial improvement in their efficiency. The proposition of algorithms that hybridize features of mixed-integer programming and heuristics (also called MIP heuristics or matheuristics) is a promising research area.

Fanjul-Peyro and Ruiz (2011) proposed a simple and powerful MIP heuristics named size-reduction (SR) algorithm. The basic idea of this method is to consider only a subset of the original set of decision variables, aiming to achieve high-quality solutions in a reduced amount of time. This paper approaches the well-known unrelated parallel machine scheduling problem, and the reduction of decision variable criterion is based on the processing times. In general, the decision variables related to the longest processing times in the available machines are set as equals zero. The SR algorithm has been applied to other scheduling problems achieving excellent results (Fanjul-Peyro et al. 2017).

For the problem under study, there is a single machine, and there is no choice for excluding decision variables according to the processing times. However, since *UB* is a pessimistic upper bound for the number of required planning periods, decision variables in the last periods could be excluded without significant losses in the achieved solution. As the objective function is the makespan minimization, the model will allocate the jobs in the first planning periods.

A probabilistic criterion to fix decision variables has been adopted based on the well-known simulated annealing algorithm (Kirkpatrick 1984). The probability of fixing decision variables as zero is calculated by the difference between the upper bound, expressed by Eq. (14), and the lower bound expressed by Eq. (13) divided by the number of the current planning period. The probability for the reduction of the problem increases with the number of periods because solutions with high quality will hardly present several jobs in the last planning periods, taking the pessimistic upper bound *UB* into consideration.

The reduced subproblem obtained by the fixed decision variables tends to be solved with a computational effort which is lower than the pure MILP model. However, there is no guarantee that the optimal solution for the reduced problem is the global optimal solution for the original problem.

Algorithm 1 presents the pseudocode for the proposed matheuristics. For each binary decision variable x_{ij}, a uniform random number between 0 and 1 (RN) is generated. If this value is lower than the difference between the upper bound and the lower bound divided by the number of the current planning periods, and if the exclusion of decision variable x_{ij} still produces a feasible solution, decision variable x_{ij} is fixed as zero. Finally, the reduced model is solved by a commercial solver.

Algorithm 1 Proposed matheuristics

```
1: procedure SR($t, n, UB, LB$)
2:     for $i <= t$ do
3:         for $j <= n$ do
4:             $RN \leftarrow U(0,1)$
5:             if $RN \leq e^{-P\times(UB-LB)/i}$ and resultant solution is feasible then
6:                 $x_{ij} \leftarrow 0$
7:             end if
8:         end for
9:     end for
10: solve the problem with commercial solver
11: end procedure
```

4 Computational results

4.1 Statistics used in the analysis of the computational experiments

The statistic used in the analysis of the computational experiments is the relative percentage deviation (RPD), which is the gap between the evaluated method ($\text{sol}_i k$) and the lower bound solution (LB_i), as presented in Eq. (15). Value sol_{ik} means the solution obtained by method k run on instance i, and LB_i denotes the lower bound solution for instance i. For each class of instances, the average relative percentage deviation (ARPD) is calculated, among all instances of each class.

$$\text{RPD}_{ik} = \frac{\text{sol}_{ik} - \text{LB}_i}{\text{LB}_i} \cdot 100. \tag{15}$$

In this work, the average computational times for the eighteen constructive heuristics are not reported since the characteristics of the proposed algorithms should not imply differences with statistical significance. For these algorithms, the computational times are negligible (less than 1 second for all the evaluated tested instances). For the two local search algorithms, the matheuristics and the MILP model, the computational times are observed.

4.2 Proposed test instances

As the SMPRC was not previously reported in the observed literature, a set of test instances is proposed, taking similar variants of the single-machine scheduling problem, presented by Low et al. (2010b) as well as Perez-Gonzalez and Framinan (2018), into consideration.

The number of jobs (n) was used as a parameter for the computational experiments. Test instances were generated for the following values: $n \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 150, 200, 250, 300\}$, as presented by

Perez-Gonzalez and Framinan (2018). 10 instances were randomly generated for each type of instance. The maximum duration of periods *P* and the maximum amount of resources for each production period *R* were generated with random values uniformly distributed between 150 and 200. The processing times and the resource consumption were generated with random values uniformly distributed between 50 and 100. The test instances are available in the following link.

4.3 Effect of phases on constructive heuristics

With the purpose of presenting which are the main effects of the phases in the obtained values of the objective function, a full factorial design (Montgomery 2017) was performed with the results of constructive heuristics, taking three factors into consideration. The first phase is composed of three factors (Sum, Avg, and Max), the second phase is also composed of three factors (LPT, A-SHARP and HILO) and the third phase is composed of two factors (FF and BF).

Figure 5 illustrates the main effects of the phases in the proposed constructive heuristics. It is possible to observe that the performance of the constructive heuristics is most strongly affected by phase 2 with the LPT priority rule.

Therefore, the optimal experimental design for the three considered phases in the constructive solution is described as follows. In the first phase, the jobs should be aggregated by the maximum value between *p* and *r*. In the second phase, the jobs should be sequenced with the LPT rule. Finally, in phase 3, the best allocation policy is the first fit. It is a good first approximation, although it is not known if there

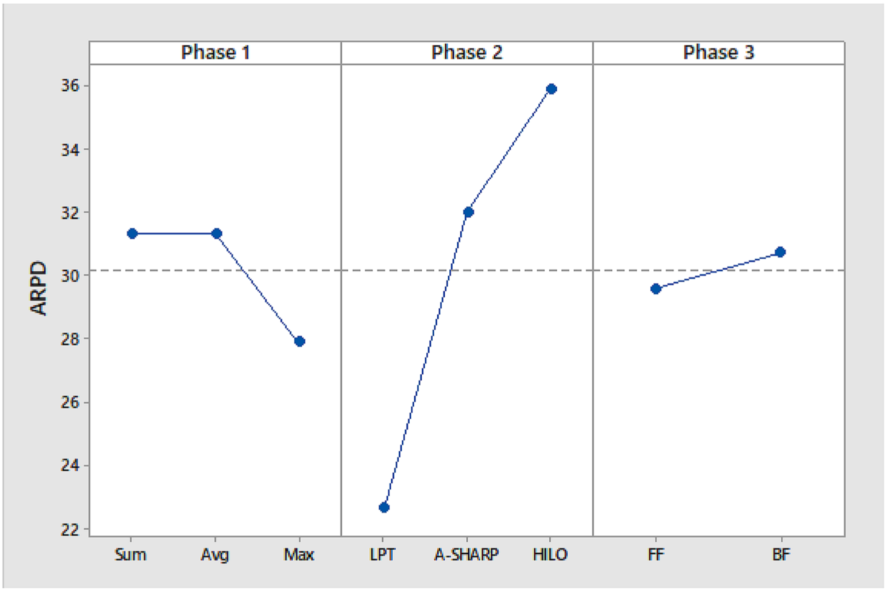

Fig. 5 Effect plots for ARPD in the phases for the constructive heuristics

will be significant differences, especially in phase 3, between FF and BF, as it can be observed as follows.

4.4 Results and discussion

All the constructive heuristics, the local search heuristics, as well as the matheuristics, were implemented using Python with Spyder IDE (https://www.anaconda.com/). For the pure MILP model, as well as the proposed matheuristics, the commercial solver used was the IBM ILOG CPLEX (https://www.ibm.com/products/ilog-cplex-optimization-studio) version 12.8. The computational experience was performed on a PC with Intel Core i7-8700 CPU 3.20 GHz × 12 and 32 GB memory. The operating system is Ubuntu 18.04.1 LTS. For the MILP and matheuristics, a time limit of 1800 s was adopted.

Table 2 describes the results for all the evaluated methods for each instance class. One can observe that the local search approaches, the matheuristics, and the MILP model present the best average RPD. Figure 6 graphically summarizes the results of Table 2 in a heat map. Dark tones represent minor ARPD values.

The constructive heuristics SFFA, AFFA, MFFA, SFFHILO, AFFHILO, MFFHILO, SBFA, ABFA, MBFA, SBFHILO, ABFHILO, and MBFHILO return the worst results, with ARPD values between 30 and 40%. The constructive heuristics SFFD, AFFD, MBFD, SBFD, and ABFD return ARPD values lower than 30%. Finally, the priority rule MFFD returns an ARPD value slightly larger than 22.0%. The local search algorithms LSFF and LSBF return ARPD values of 20.97% and 21.35%, respectively. The proposed MILP model returns an ARPD of 24,71%. Finally, the proposed matheuristics returns an ARPD of 19.55%, the best average results for all the evaluated methods.

To validate the results, it is important to verify whether the previous differences in the RPD values are statistically significant. An analysis of variance (ANOVA) (Montgomery 2017) has been applied. The p value is very close to zero. It is possible to see in Fig. 7 the mean plots with confidence intervals ($\alpha = 0.05$) of all methods proposed. It has been verified that there are statistically remarkable differences between the average RPD values and some methods proposed.

Figure 7 presents the 95% confidence interval for ARPD in each method. Figure 8 illustrates the same results for the 4 best methods. To evaluate the distribution of the results for the main considered methods for each instance class, the 95% confidence interval for ARPD values in Fig. 9 has been summarized.

One can observe that the matheuristics, in general lines, returns better or equal results than the MILP model in the evaluated instances. This evidence indicates that the utilization of random criteria embedded with exact methods can lead to improvements in the deterministic method. In addition, the local search methods are better than MIP and MILP methods in the small-sized test problems and worse in the large-sized test problems.

Concerning the computational times, the following results can be observed. Since the constructive heuristics are deterministic and their computational times were negligible (less than 1 s), they were not considered in this analysis. The two local

Table 2 Average RPD for each method in each instance size

Method	Size														Total
	10	20	30	40	50	60	70	80	90	100	150	200	250	300	
SFFD	38.37	38.36	24.53	21.48	30.76	23.23	19.43	22.17	16.63	18.17	18.47	14.87	16.05	14.29	22.63
AFFD	38.37	38.36	24.53	21.48	30.76	23.23	19.43	22.17	16.63	18.17	18.47	14.87	16.05	14.29	22.63
MFFD	35.34	38.68	24.82	20.39	31.13	23.42	18.83	21.82	16.34	18.03	18.33	14.21	15.87	13.76	22.21
SFFA	39.9	44.2	32.16	28.63	41.79	34.09	30.12	30.86	26.77	30	29.38	27.16	28.2	26.83	32.15
AFFA	39.9	44.2	32.16	28.63	41.79	34.09	30.12	30.86	26.77	30	29.38	27.16	28.2	26.83	32.15
MFFA	41.93	43.25	32.46	28.3	41.12	35.91	28.25	31.52	27.22	30.7	30.29	27.2	28.17	27.44	32.41
SFFHILO	42.16	47.3	37.25	35.39	47.99	41.07	39.35	39.67	35.29	37.83	40.05	36.61	40.2	38.50	39.90
AFFHILO	42.16	47.3	37.25	35.39	47.99	41.07	39.35	39.67	35.29	37.83	40.05	36.61	40.2	38.50	39.90
MFFHILO	46.49	50.27	38.68	35	44.88	38.19	33.63	38.32	30.69	35.68	35.7	33.12	33.87	32.57	37.65
SBFD	38.93	38.36	24.53	21.63	30.95	23.23	19.43	22.17	16.63	18.19	18.45	14.9	16.13	14.27	22.70
ABFD	38.93	38.36	24.53	21.63	30.95	23.23	19.43	22.17	16.63	18.19	18.45	14.9	16.13	14.27	22.70
MBFD	35.64	39.52	25.09	21.36	32.09	24.08	19.42	22.8	16.47	18.44	19.05	15.63	16.68	14.85	22.94
SBFA	39.9	44.25	30.35	29.24	41.27	33.85	29.82	30.67	25.93	29.77	28.92	26.34	28.14	26.59	31.79
ABFA	39.9	44.25	30.35	29.24	41.27	33.85	29.82	30.67	25.93	29.77	28.92	26.34	28.14	26.59	31.79
MBFA	41.2	43.77	33.3	27.83	40.1	34.72	27.8	31.13	26.6	30.29	28.94	25.29	26.75	26.32	31.72
SBFHILO	43.21	47.92	37	34.67	46.49	37.39	37.66	38.17	34.13	35.48	38.24	34.33	39.37	37.70	38.70
ABFHILO	43.21	47.92	37	34.67	46.49	37.39	37.66	38.17	34.13	35.48	38.24	34.33	39.37	37.70	38.70
MBFHILO	45.43	49.01	38.68	33.54	43.65	37.1	32.43	36.61	29.04	34.46	34.18	31.09	31.34	30.75	36.24
LSFF	29.45	35.9	22.98	19.45	30.06	22.23	17.95	21.06	15.85	17.61	17.96	13.79	15.62	13.64	20.97
LSBF	30.1	35.54	22.94	19.78	30.06	22.34	18.51	21.61	15.71	17.81	18.65	15.02	16.22	14.61	21.35
MIP	29.37	34.96	22.02	18.22	29.27	24.08	14.98	17.46	14.7	15.95	15.47	12.28	12.72	12.29	19.55
MILP	41.91	42.6	26.18	21.18	31.25	23.98	19.49	21.74	16.45	18.11	19.41	19.28	22.58	21.89	24.71

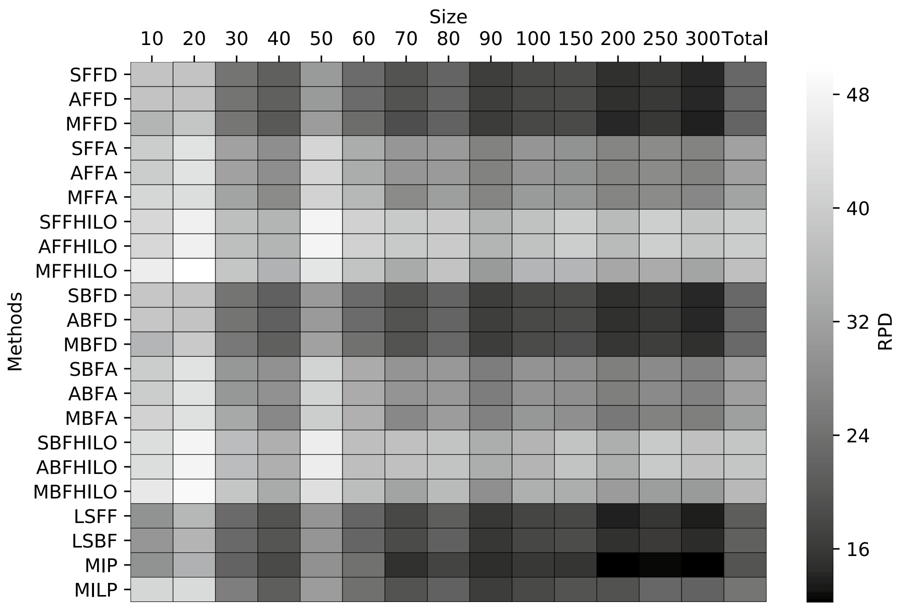

Fig. 6 Heat map for results for each method in each instance size

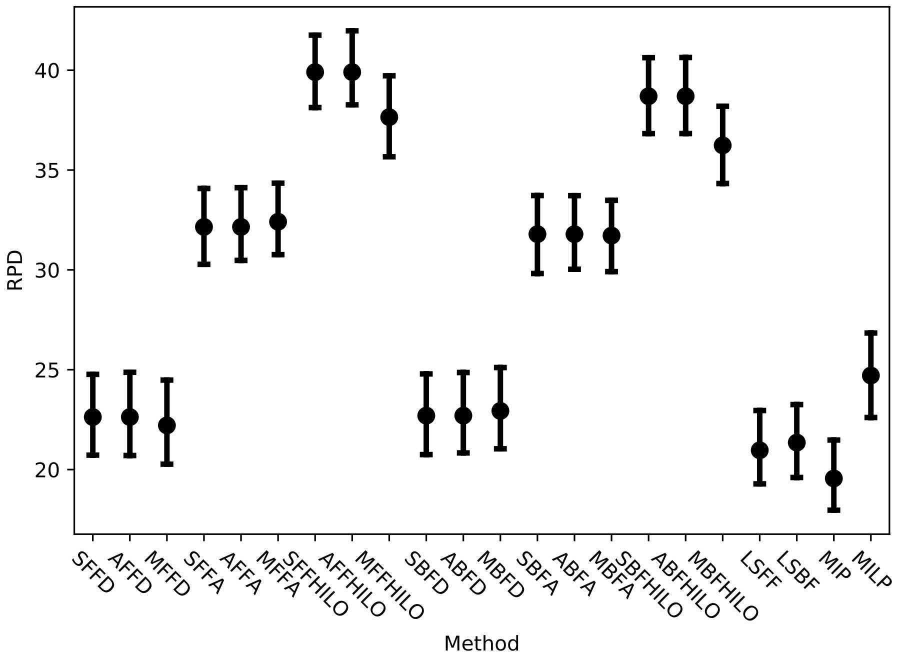

Fig. 7 95% confidence interval for ARPD for each method

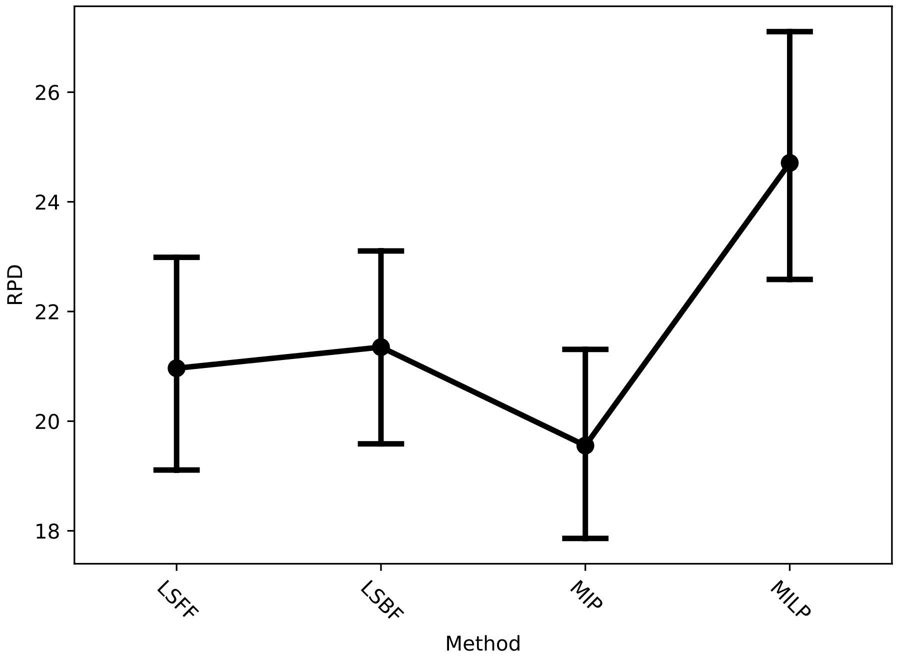

Fig. 8 95% confidence interval for ARPD for the best methods

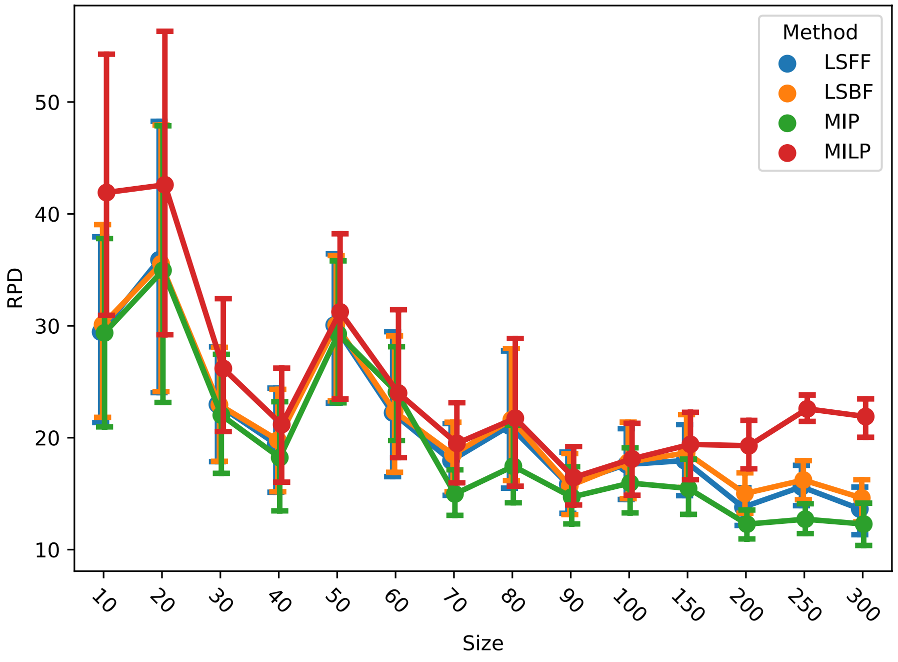

Fig. 9 95% confidence interval for the ARPD variable for the best methods for each instance size

search approaches, as well as the matheuristics, are run ten times for each instance. The MILP model, as a deterministic method, is run a single time. Table 3 presents the average computational times for each class of instances. In this table, the column "Total" indicates the general average of computational times for each considered method.

The local search algorithms present a computational effort which is significantly lower than the MIP and MILP methods. With regard to the matheuristics, this algorithm, in general, returns better results than the MILP model with a lower computational effort. Figure 10 presents a confidence interval ($\alpha = 0.05$) for the average computational times in each class of instances. The above-mentioned figure is divided into two parts illustrating the local search algorithms and the exact-based approaches. One can observe that the difference between scale is significant between the two charts.

For the local search algorithms, the computational time growth is exponential. There is no significant difference between FF and BF strategies. The single difference between the two local search methods occurs in the class of instances higher than 150 jobs, in which the LSFF algorithm presents slightly lower values for the computational times than the LSBF method. Such similar behavior can be explained by the construction phase based on the LPT rules, as illustrated in Fig. 5.

Taking the exact-based approaches into consideration, it is possible to observe a remarkable discrepancy in the presented results, mainly in the medium-sized instances. In the majority of evaluated instances, the matheuristics returned the solutions with lower computational times than the proposed MILP model. Furthermore, the matheuristics returned solutions with better objective function values, outperforming the MILP model.

5 Final remarks

In this paper, a new variant for the single-machine scheduling problem has been investigated considering processing times and resource consumption. The objective function is the makespan minimization. Eighteen constructive heuristic algorithms based on priority rules and bin packing policies have been developed. Two local search algorithms and a matheuristics based on the size-reduction heuristics and the simulated annealing algorithm have also been proposed.

Computational experiments have been carried out to evaluate the performance of the proposed algorithms as well as the developed MILP model. The relative deviation statistic has been used as the performance measure. In general lines, for the most tested problem instances, the constructive algorithms based on first fit A-Sharp strategies generate the worst solutions. On the other hand, the constructive algorithms based on the best fit decreasing policies outperformed all other tested priority rules in most cases. In practical terms, the LSFF and LSBF heuristics can be used in an industrial environment because they present a good trade-off between the quality of the achieved solutions and the computational effort.

The two local search heuristics, the MILP model, and the matheuristics present similar average results, although the local search algorithms present a

Table 3 Average computational times for each method for each instance size

Methods	Size														Total
	10	20	30	40	50	60	70	80	90	100	150	200	250	300	
LSFF	0.00	0.00	0.01	0.01	0.02	0.03	0.05	0.08	0.10	0.12	0.40	0.93	1.84	2.97	0.47
LSBF	0.00	0.00	0.01	0.01	0.04	0.06	0.08	0.11	0.16	0.20	0.65	1.48	3.41	4.64	0.78
MIP	0.22	190.85	187.03	480.32	401.31	654.89	1311.20	1810.69	1802.08	1801.03	1802.14	1803.88	1807.02	1809.50	1133.01
MILP	0.26	3.98	109.81	1554.83	1801.21	1801.53	1801.25	1800.27	1800.94	1800.89	1141.10	1455.17	1800.15	1800.30	1326.91

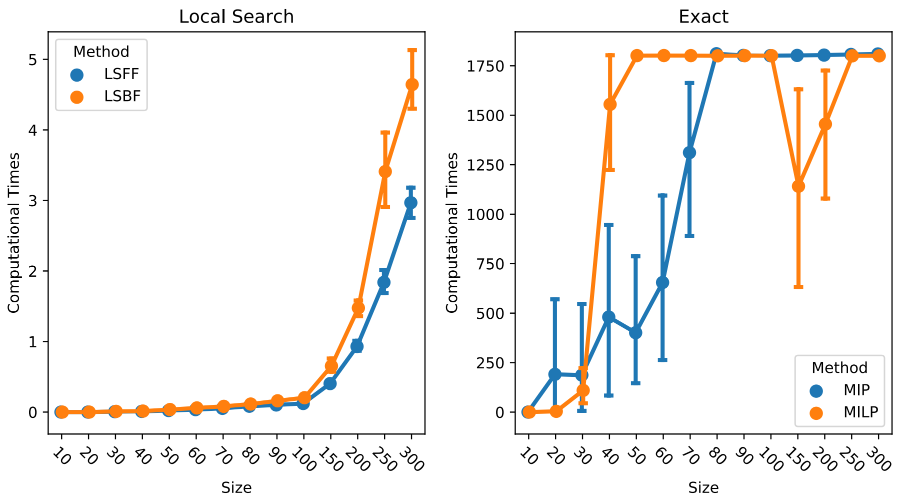

Fig. 10 95% confidence interval for computational times for the best methods for each instance size

lower computational effort. Concerning the quality of the generated solutions, the matheuristics presents better average results, although the difference between LSFF, LSBF, and MILP methods is not significant.

As extensions of this work, the use of metaheuristics is recommended to improve the solutions generated by the constructive heuristics. In addition, the mathematical formulation, as well as the proposed heuristics, could be generalized for any quantity of criteria. Another possibility is the consideration of different objective functions for the variant under proposition. Finally, considering the practical interest of resource consumption in scheduling problems, periodical resource constraints could be extended to other production environments such as parallel machines, flow shop, job shop, and open shop.

Acknowledgements This study was financed in part by the Coordination for the Improvement of Higher Education Personnel (CAPES)- Finance Code 001. This work was partially funded by National Council for Scientific and Technological Development (CNPq) through grants 404232/2016-7 and 303594/2018-7.

References

Afzalirad M, Rezaeian J (2016) Resource-constrained unrelated parallel machine scheduling problem with sequence dependent setup times, precedence constraints and machine eligibility restrictions. Comput Ind Eng 98:40–52. https://doi.org/10.1016/j.cie.2016.05.020

Afzalirad M, Shafipour M (2018) Design of an efficient genetic algorithm for resource-constrained unrelated parallel machine scheduling problem with machine eligibility restrictions. J Intell Manuf 29(2):423–437. https://doi.org/10.1007/s10845-015-1117-6

Ángel Bello F, Álvarez A, Pacheco J, Martínez I (2011) A heuristic approach for a scheduling problem with periodic maintenance and sequence-dependent setup times. Comput Math Appl 61(4):797–808. https://doi.org/10.1016/j.camwa.2010.12.028

Birgin EG, Ronconi DP (2012) Heuristic methods for the single machine scheduling problem with different ready times and a common due date. Eng Optim 44(10):1197–1208. https://doi.org/10.1080/0305215X.2011.634409

Edis EB, Ozkarahan I (2012) Solution approaches for a real-life resource-constrained parallel machine scheduling problem. Int J Adv Manuf Technol 58(9):1141–1153. https://doi.org/10.1007/s00170-011-3454-8

Fanjul-Peyro L, Ruiz R (2011) Size-reduction heuristics for the unrelated parallel machines scheduling problem. Comput Oper Res 38(1):301–309. https://doi.org/10.1016/j.cor.2010.05.005

Fanjul-Peyro L, Perea F, Ruiz R (2017) Models and matheuristics for the unrelated parallel machine scheduling problem with additional resources. Eur J Oper Res 260(2):482–493. https://doi.org/10.1016/j.ejor.2017.01.002

Framinan JM, Leisten R, Ruiz R (2014) Manufacturing scheduling systems. Springer, Berlin

Hsu CJ, Low C, Su CT (2010) A single-machine scheduling problem with maintenance activities to minimize makespan. Appl Math Comput 215(11):3929–3935. https://doi.org/10.1016/j.amc.2009.11.040

Ji M, He Y, Cheng T (2007) Single-machine scheduling with periodic maintenance to minimize makespan. Comput Oper Res 34(6):1764–1770. https://doi.org/10.1016/j.cor.2005.05.034 (part Special Issue: Odysseus 2003 Second International Workshop on Freight Transportation Logistics)

Ji M, Wang JY, Lee WC (2013) Minimizing resource consumption on uniform parallel machines with a bound on makespan. Comput Oper Res 40(12):2970–2974. https://doi.org/10.1016/j.cor.2013.06.011

Karhi S, Shabtay D (2018) Single machine scheduling to minimise resource consumption cost with a bound on scheduling plus due date assignment penalties. Int J Prod Res 56(9):3080–3096. https://doi.org/10.1080/00207543.2017.1400708

Kirkpatrick S (1984) Optimization by simulated annealing: quantitative studies. J Stat Phys 34(5–6):975–986

Low C, Hsu CJ, Su CT (2010a) A modified particle swarm optimization algorithm for a single-machine scheduling problem with periodic maintenance. Expert Syst Appl 37(9):6429–6434. https://doi.org/10.1016/j.eswa.2010.02.075

Low C, Ji M, Hsu CJ, Su CT (2010b) Minimizing the makespan in a single machine scheduling problems with flexible and periodic maintenance. Appl Math Model 34(2):334–342. https://doi.org/10.1016/j.apm.2009.04.014

Martello S (1990) Knapsack problems: algorithms and computer implementations. Wiley-Interscience series in discrete mathematics and optimization

Montgomery DC (2017) Design and analysis of experiments. Wiley, New York

Pacheco J, Ángel-Bello F, Álvarez A (2013) A multi-start tabu search method for a single-machine scheduling problem with periodic maintenance and sequence-dependent set-up times. J Sched 16(6):661–673. https://doi.org/10.1007/s10951-012-0280-2

Perez-Gonzalez P, Framinan JM (2018) Single machine scheduling with periodic machine availability. Comput Ind Eng 123:180–188. https://doi.org/10.1016/j.cie.2018.06.025

Pitombeira-Neto AR, Prata BdA (2019) A matheuristic algorithm for the one-dimensional cutting stock and scheduling problem with heterogeneous orders. TOP. https://doi.org/10.1007/s11750-019-00531-3

Prata BA, Pitombeira-Neto AR, Sales CJM (2015) An integer linear programming model for the multiperiod production planning of precast concrete beams. J Constr Eng Manag 141(10):04015029. https://doi.org/10.1061/(ASCE)CO.1943-7862.0000991

Scheithauer G (2017) Introduction to cutting and packing optimization: problems, modeling approaches, solution methods, vol 263. Springer, Berlin

Valente JM (2007) Heuristics for the single machine scheduling problem with early and quadratic tardy penalties. Eur J Ind Eng 1(4):431–448

Ventura JA, Kim D (2003) Parallel machine scheduling with earliness-tardiness penalties and additional resource constraints. Comput Oper Res 30(13):1945–1958. https://doi.org/10.1016/S0305-0548(02)00118-1

Villa F, Vallada E, Fanjul-Peyro L (2018) Heuristic algorithms for the unrelated parallel machine scheduling problem with one scarce additional resource. Expert Syst Appl 93:28–38. https://doi.org/10.1016/j.eswa.2017.09.054

Wang JB, Wang MZ (2012) Single-machine scheduling to minimize total convex resource consumption with a constraint on total weighted flow time. Comput Oper Res 39(3):492–497. https://doi.org/10.1016/j.cor.2011.05.026

Wang XR, Wang JJ (2013) Single-machine scheduling with convex resource dependent processing times and deteriorating jobs. Appl Math Model 37(4):2388–2393. https://doi.org/10.1016/j.apm.2012.05.025

Wu L, Cheng CD (2016) On single machine scheduling with resource constraint. J Combin Optim 31(2):491–505. https://doi.org/10.1007/s10878-014-9721-5

Yeh WC, Chuang MC, Lee WC (2015) Uniform parallel machine scheduling with resource consumption constraint. Appl Math Model 39(8):2131–2138. https://doi.org/10.1016/j.apm.2014.10.012

Yu X, Zhang Y, Steiner G (2014) Single-machine scheduling with periodic maintenance to minimize makespan revisited. J Sched 17(3):263–270. https://doi.org/10.1007/s10951-013-0350-0

Zheng XL, Wang L (2016) A two-stage adaptive fruit fly optimization algorithm for unrelated parallel machine scheduling problem with additional resource constraints. Expert Syst Appl 65:28–39. https://doi.org/10.1016/j.eswa.2016.08.039

Zheng XL, Wang L (2018) A collaborative multiobjective fruit fly optimization algorithm for the resource constrained unrelated parallel machine green scheduling problem. IEEE Trans Syst Man Cybern Syst 48(5):790–800. https://doi.org/10.1109/TSMC.2016.2616347

Zhu Z, Chu F, Sun L, Liu M (2013) Single machine scheduling with resource allocation and learning effect considering the rate-modifying activity. Appl Math Model 37(7):5371–5380. https://doi.org/10.1016/j.apm.2012.09.072