RAIRO-Oper. Res. 55 (2021) 2491–2524
https://doi.org/10.1051/ro/2021107

RAIRO Operations Research
www.rairo-ro.org

THE INTEGRATED CUTTING AND PACKING HETEROGENEOUS PRECAST BEAMS MULTIPERIOD PRODUCTION PLANNING PROBLEM

Kennedy Anderson Gumarães de Araújo[1], Tibérius de Oliveira e Bonates[2] and Bruno de Athayde Prata[3,*]

Abstract. We introduce a novel variant of cutting production planning problems named Integrated Cutting and Packing Heterogeneous Precast Beams Multiperiod Production Planning (ICP-HPBMPP). We propose an integer linear programming model for the ICP-HPBMPP, as well as a lower bound for its optimal objective function value, which is empirically shown to be closer to the optimal solution value than the bound obtained from the linear relaxation of the model. We also propose a genetic algorithm approach for the ICP-HPBMPP as an alternative solution method. We discuss computational experiments and propose a parameterization for the genetic algorithm using D-optimal experimental design. We observe good performance of the exact approach when solving small-sized instances, although there are difficulties in finding optimal solutions for medium and large-sized problems, or even in finding feasible solutions for large instances. On the other hand, the genetic algorithm is shown to typically find good-quality solutions for large-sized instances within short computing times.

Mathematics Subject Classification. 90C27, 90B30, 90C59, 62P30.

Received March 8, 2021. Accepted July 20, 2021.

1. Introduction

Nowadays, concrete precast production is increasingly trending in constructions sites. There are great advantages in using such kind of production, such as better and cheaper elements, as well as the potential to severely shorten construction time as compared to conventional methods. The precast element we consider in this work is a concrete precast beam, which is a kind of beam that is cast in plants away from the construction site, in a controlled environment.

These beams are heterogeneous in the sense that they can vary with respect to curing time, length and the number of traction elements used. We refer to the problem of planning the production of such beams to fulfill the clients demand within a given time horizon as the Heterogeneous Precast Beams Multiperiod Production Planning Problem (HPBMPP).

Keywords. Precast beams, modular construction, integer linear programming, metaheuristics, genetic algorithms.

[1] Department of Applied Mathematics, University of São Paulo, São Paulo, Brazil
[2] Department of Statistics and Applied Mathematics, Federal University of Ceará, Fortaleza, Brazil
[3] Department of Industrial Engineering, Federal University of Ceará, Fortaleza, Brazil.
*Corresponding author: baprata@ufc.br

In [1], the authors proposed four integer programming models for the HPBMPP, considering prestressed precast beams instead of conventional concrete precast beams. One of the proposed models minimizes the total idle capacity in the molds along the time horizon, two models minimize the production makespan, and one model minimizes the total completion time. The authors also proposed several solution methods, in particular a size reduction heuristic that succeeded in finding high-quality solutions in shorter time and using less memory compared to exact methods.

In this work, we propose a variant model of the HPBMPP, which consists in the integration of the production of bars, which are used in the precast beam production, into the problem. We divide the bars into two groups: *standard bars* and *leftovers*. Standard bars are new bars of standardized lengths, and leftovers are a type of bar that cannot be readily used in the beam production but can be stored in stock to produce other bars in the future. In this study, we consider that both standard bars and leftovers vary with respect to length. The production of bars to be used in the beam production can be made by the cutting of standard bars or leftovers in stock, or by the process of cutting overlapping leftovers. The overlapping process consists in merging two or more leftovers in order to create a larger bar that can be cut to produce a bar of appropriate length that can be used in beam production. In this work, we only consider overlapping of two bars. To the best of our knowledge, the consideration of overlapping bars has not been previously studied.

We consider the integration into a single production planning problem of the cutting process of bars, or of overlapping bars, which must be packed in the molds for the production of a given demand of beams. We refer to this problem as the Integrated Cutting and Packing Heterogeneous Precast Beams Multiperiod Production Planning Problem (ICP-HPBMPP). Note that in this work we consider beams that are not prestressed. The mathematical model we propose is based on the model by [2], which deals with the cutting stock/leftover problem, and on the model by [1] for the HPBMPP. We consider that the bars needed to supply the beam production can be produced by cutting bars or leftovers in stock or by overlapping leftovers in stock. The stock is static, *i.e.*, we are given an initial stock that is not replenished over the entire time horizon.

The ICP-HPBMPP is of practical interest because optimizing the production of prestressed beams has the potential effect of speeding up overall construction time, while improving the usage of molds and bar stock, while minimizing bars loss. An economical usage of bar stock may result in a reduction of unused bars in the construction site, which can improve production flow. Furthermore, the reduction of concrete and bar loss may lead to a positive impact in the environment. An optimized process also allows factories to accept additional orders due to shorter lead times. Moreover, the production cost with an optimized process will be lower, which may lead to a reduction of the final product's price, increasing competitiveness.

It is argued in [1] that the HPBMPP is NP-hard since it includes, as a particular case, the classical one-dimensional cutting stock problem. Thus, the HPBMPP can become too difficult to solve as the dimension of instances increases. The computational results reported in Section 6 show that the ICP-HPBMPP can be difficult to solve to optimality, justifying the use of decomposition techniques and heuristic procedures to deal with the problem. This also suggests that the HPBMPP is interesting to be studied from a theoretical point of view.

The remainder of this paper is organized as follows. In Section 2, we discuss the literature on similar problems to the ICP-HPBMPP. In Section 3, we formally define the problem, propose an integer linear programming model for its solution, argue about its NP-hardness, and propose a lower bound for its optimal objective function value. In Section 4, we present three constraint programming models for the generation of packing, cutting and overlapping patterns. In Section 5, we propose a genetic algorithm for the problem under study. In Section 6, we discuss several computational experiments conducted with artificially generated instances and discuss the results of the proposed solution methods. In Section 7, we discuss the conclusions and contributions of this paper, point out research gaps, and suggest future work.

2. Literature review

To the best of our knowledge the ICP-HPBMPP is not defined in the literature, even though the problem has similarities with one-dimensional cutting stock problems (1DCSP) and one-dimensional packing problems (1DPP). On the order hand, 1DCSP, 1DPP, and their variants have been substantially studied in the literature.

As far as one-dimensional cutting and packing problems (C&P) are concerned, the studies of [10,11] proposed a column generation algorithm to solve the linear relaxation of large instances of 1DCSP. Such studies served as basis for a number of subsequent works. In [18], a heuristic for the 1DCSP was proposed based on the solution of the linear relaxation supplemented by a one-pass branching up procedure. The authors validated the proposed heuristic approach, testing on benchmark instances and on a case study. In [8], the authors introduced a typology of C&P problems, unifying notions in the literature to guide further research on particular types of those problems. Two branch-and-price approaches were proposed in [21] to find optimal solutions for the 1DCSP. In [24], a new typology was presented to categorize the types of C&P problems in the literature between the years 1995 and 2004, introducing new categorization criteria. A model was proposed in [20] for the multiperiod one-dimensional cutting stock problems (M1DCSP), considering the use of objects/leftovers in stock. In [16], an integer linear model for the M1DCSP was proposed, a column generation procedure was implemented to solve the linear relaxation, and two rounding heuristics were developed for finding integer solutions to the problem. In [13], a mathematical model for the general integrated lot-sizing and cutting stock problem was proposed; additionally, a vast classification of the literature on that problem was performed, providing directions for future research.

Regarding the C&P problems and optimization approaches in precast production, De Castilho *et al.* [7] described the problem of minimizing production costs for slabs of precast prestressed concrete joists and introduced a genetic algorithm to solve it. An integer linear programming model for multiperiod production planning of precast concrete beams was proposed in [3], which can be seen as a special case of the HPBMPP. In [2], the authors introduced a mathematical model for the cutting stock/leftover problem and suggested a column generation technique for finding the problem's linear relaxation solution. In [22], a mathematical model was proposed based on the multiperiod cutting stock problem for the production planning problem of joists in trusses slabs industries. The authors suggested a solution method based on column generation to solve the linear relaxation of the problem. In [1], several integer linear programming models for the Heterogeneous Prestressed Precast Beams Multiperiod Production Planning Problem were proposed. The authros established the NP-hardness of the problem and suggested a constraint programming model for generating cutting patterns for the problem. The authors also carried out computational experiments to validate the performance of the integer linear programming models. In [23], the authors introduced a two-hierarchy simulation-genetic algorithm hybrid model for precast production to ensure the on-time delivery of precast components minimizing the production cost, while simultaneously optimizing the resource waste under uncertainty in the processing time of each operation. The model was validated by means of a case study.

The problem which we study in this work is the integration of the cutting stock/leftover problem proposed in [2] and the HPBMPP introduced in [1]. We explore its solution *via* exact methods and heuristics methods in the case where instances cannot be solved by the state-of-art solvers.

3. Problem statement

In this section, we formally define the ICP-HPBMPP and propose an integer linear programming model for its solution based on the models proposed in [2] for the Cutting Stock/Leftover Problem (CSLP) and in [1] for the Heterogeneous Prestressed Precast Beams Multiperiod Production Planning Problem (HPPBMPP).

The ICP-HPBMPP consists in finding a feasible production planning to cast certain quantities of prestressed precast concrete beams, possibly of different types, while minimizing the total length of pieces of bars that cannot be used as *leftover*. A leftover is understood here as a piece of bar that can be cut or overlapped in the

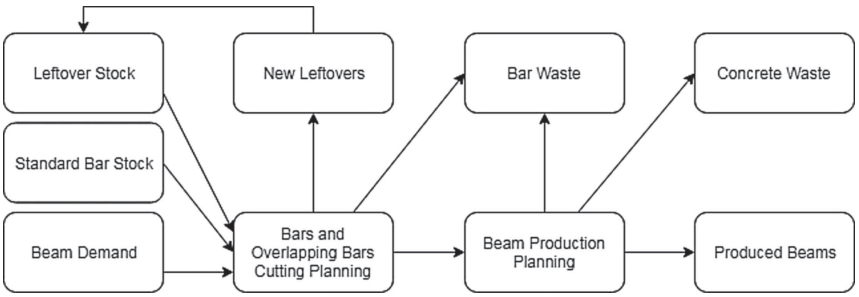

FIGURE 1. Cutting and packing production flowchart.

future to meet new demands and is not considered waste. The beam factory has a fixed amount of bars and bar leftovers with standard lengths in stock that can be used within a given time horizon.

Each mold can only be used to cast one type of beam at a time. It is possible, however, to simultaneously cast beams of different lengths in the same mold, as long as they are of the same type. The total length of the beams produced during a given period in a given mold cannot be greater than the mold's capacity, and the total number of days required to complete the entire production cannot be greater than a given time horizon. After the process of cutting bars is finished, they are packed in molds in order to produce the beams. Note that different beam types can demand different numbers of bars. For this reason, we refer to this problem as a Cutting and Packing problem. The ICP-HPBMPP process can be seen in Figure 1.

As input of the problem we have a deterministic, static demand of beams, with their respective types and lengths, stock of bars and stock of bars leftovers, with their respective lengths. The cutting planning of bars is made for the entire time horizon, resulting in more bars leftovers (which can be used in another production planning) and, possibly, incurring in bar loss. The bars that are cut are then packed in the molds for the beam production, along the given time horizon. After the production of all demanded beams is met, there will usually be concrete waste of the beams and additional loss of bars.

3.1. Integer linear programming model

In order to define a model for the ICP-HPBMPP, we make use of the same parameters defined in [1], as follows:

- M: number of molds in which the beams are produced;
- T: number of available periods to complete the production;
- C: number of beam types;
- q_c: number of distinct lengths of beams of type c, with $c = 1, \ldots, C$;
- $l(c,1), \ldots, l(c,q_c)$: real numbers corresponding to the actual lengths of beams of type c, with $c = 1, \ldots, C$;
- $d(c,k)$: demand for beams of type c and length $l(c,k)$, with $c = 1, \ldots, C$ and $k = 1, \ldots, q_c$;
- t_c: integer number corresponding to the curing time (in terms of periods) of beams of type c, for $c = 1, \ldots, C$;
- L_m: real number corresponding to the capacity of mold m, with $m = 1, \ldots, M$;
- $P_i = \left(c_i, \left(a^i_1, \ldots, a^i_{q_{c_i}}\right)\right)$: packing pattern, where c_i stands for the beam type associated with pattern P_i and $a^i_1, \ldots, a^i_{q_{c_i}}$ represent the quantity of each beam of length $l(c_i,1), \ldots, l(c,q_{c_i})$ in patterns P_i, with $i = 1, \ldots, r$, $c_i = 1, \ldots, C$. Note that r represents the number of packing patterns;
- P_0: special pattern, which is used to denote that a mold is currently being used for the casting of a pattern that began in a previous period and whose production extends at least up to the current period.

Note that an idle mold (in other words, a mold that is not being used during a specific period) is not assigned the pattern P_0. In fact, it has no pattern assigned to it.

In order to refer to specific information on a given pattern $P_i = \big(c_i, (a_1, \ldots, a_{q_{c_i}})\big)$, we define the following notation:

- $\mathcal{N}_i(c,k)$: number of beams of type c and length $l(c,k)$ that pattern P_i includes. If $c = c_i$, then $\mathcal{N}_i(c,k) = a_k$, with $k \in \{1, \ldots, q_{c_i}\}$; otherwise, $\mathcal{N}_i(c,k) = 0$, for any k.
- $u(P_i)$: capacity used by P_i, *i.e.*, $u(P_i) = \sum_{k=1}^{q_{c_i}} l(c_i,k) \cdot P_i(c_i,k)$, with $i = 1, \ldots, r$.
- E_i: number of periods required to produce the beams in P_i, with $i = 1, \ldots, r$. This number equals the quantity of consecutive periods in which P_i remains occupying a mold and is precisely the curing time of beams of type c_i, given by t_{c_i}.

Given a set of patterns $\mathcal{P} = \{P_1, \ldots, P_r\}$, not including P_0, we define some important sets as follows:

- $Q(m)$: set containing the indices of the patterns in $\mathcal{P}$ whose capacity does not exceed the capacity of the m-th mold: $Q(m) = \{i \in \{1, \ldots, r\} : u(P_i) \leq L_m\}$, for $m = 1, \ldots, M$. Note that the same pattern can belong to $Q(m)$ and $Q(m')$, with m and m' being two different molds of potentially distinct lengths.
- $Q^\star(m) = Q(m) \cup \{0\}$;
- $S(j)$: set of indexes of the patterns that have curing time $j \in \{1, \ldots, R\}$, with $R = \max\{t_c : c = 1, \ldots, C\}$ being the largest curing time of all beam types present in the problem instance.

In what follows, we present the parameters that concern bars and bars leftover:

- W: number of different bar lengths;
- V: number of different bar leftover lengths;
- H: number of cutting patterns;
- O: number of overlapping patterns;
- Γ: number of different mold lengths;
- $b_1, \ldots, b_W$: bar lengths;
- $b_{W+1}, \ldots, b_{W+V}$: bar leftover lengths allowed. Note that this data narrows the types of cutting, and overlapping patterns;
- $\mathcal{L}_1, \ldots, \mathcal{L}_\Gamma$: mold lengths. Note that this data narrows the types of cutting, and overlapping patterns;
- $G(\mathcal{L}_\gamma)$ = set of molds which are of length $\mathcal{L}_\gamma$, $\gamma = 1, \ldots, \Gamma$;
- H_w: set of cutting patterns for bar of length b_w that do not include leftovers.
- $H_w(v)$: set of cutting patterns for bar type w that include leftovers of length b_{W+v};
- $\mathbb{O}$: set of overlapping patterns;
- $\mathbb{O}(\gamma)$: set of overlapping patterns that produce bars of length $\mathcal{L}_\gamma$.
- $I_h = \big(w_h, \big(a_1^h, \ldots, a_\Gamma^h, a_{\Gamma+1}^h, \ldots, a_{\Gamma+V}^h\big)\big)$: cutting pattern used to cut a bar of index $w_h = 1, \ldots, W+V$, with $h = 1, \ldots, H$. Note that $a_1^h, \ldots, a_\Gamma^h$ are the number of bars of lengths $\mathcal{L}_1, \ldots, \mathcal{L}_\Gamma$ and $a_{\Gamma+1}^h, \ldots, a_{\Gamma+V}^h$ are the number of bars of lengths $b_{W+1}, \ldots, b_{W+V}$;
- $\mathbb{O}_\mu = (\gamma_\mu, (a_1^\mu, \ldots, a_V^\mu))$: overlapping pattern that generates a bar of length $\mathcal{L}_{\gamma_\mu}$, with $\gamma_\mu = 1, \ldots, \Gamma$ and $\mu = 1, \ldots, O$. Note that $a_1^\mu, \ldots, a_V^\mu$ are the number of bars of lengths $b_{W+1}, \ldots, b_{W+V}$;
- D_{c_i} = number of bars that a pattern P_i with beam type c_i demands;
- e_w = number of bars of length b_w in stock, leftover or otherwise, with $w = 1, \ldots, W+V$;
- $a_{v,\mu}$ = number of leftovers of length b_{W+v} in overlapping pattern $\mathbb{O}_\mu$, with $\mu = 1, \ldots, O$.
- $a_{\gamma,h,w}$ = number of objects of length $\mathcal{L}_\gamma$ cut from a bar of length b_w following a cutting pattern I_h that generates no leftover, with with $w = 1, \ldots, W+V$;
- $a_{\gamma,h,w,v}$ = number of objects of length $\mathcal{L}_\gamma$ cut from a bar of length b_w following a cutting pattern I_h that generates a leftover of length b_{W+v}, with $w = 1, \ldots, W$ and $v = 1, \ldots, V$.
- $f_{h,w}$ = waste resulting from using a cutting pattern I_h to cut a bar of length b_w generating no leftover, with $w = 1, \ldots, W+V$.
- $f_{h,w,v}$ = waste resulting from using a cutting pattern I_h to cut a bar of length b_w generating a leftover of length b_{W+v}, with $w = 1, \ldots, W$ and $v = 1, \ldots, V$.
- f_μ = waste of bar produced by overlapping pattern $\mathbb{O}_\mu$, with $\mu = 1, \ldots, O$.

We present the decision variables below:

$$x_i^{m,t} = \begin{cases} 1, & \text{if the packing pattern } P_i \text{ starts to be used in} \\ & \text{mold } m \text{ at period } t \text{ (and its usage, naturally,} \\ & \text{lasts for } E_i \text{ periods);} \\ 0, & \text{otherwise.} \end{cases}$$

$$z_t = \begin{cases} 1, & \text{if as least one mold is used at period } t, \text{ for } t = 1, \ldots, T; \\ 0, & \text{otherwise.} \end{cases}$$

$y_{h,w}$: number of bars of length b_w cut following a cutting pattern $I_h \in H_w$.
$y_{h,w,v}$: number of bars of length w cut following a cutting pattern $I_h \in H_w(v)$ generating a leftover of length b_{W+v}.
o_μ : number of times the overlapping pattern $\mathbb{O}_\mu$ was used, $\mu \in \mathbb{O}$.

Note that variables $y_{h,w}, y_{h,w,v}$, and o_μ are nonnegative integer decision variables. We present the integer linear programming model proposed for the ICP-HPBMPP as follows:

(ICP) min

$$\begin{aligned} & \lambda_1 \sum_{t=1}^{T} z_t + \lambda_2 \sum_{w=1}^{W} \sum_{h \in H_w} f_{h,w} y_{h,w} \\ & + \lambda_3 \sum_{w=1}^{W} \sum_{v=1}^{V} \sum_{h \in H_w(v)} f_{h,w,v} y_{h,w,v} \\ & + \lambda_4 \left(\sum_{w=W+1}^{W+V} \sum_{h \in H_w} f_{h,w} y_{h,w} + \sum_{\mu \in \mathbb{O}} f_\mu o_\mu \right) \end{aligned} \tag{3.1}$$

s.t.

$$\sum_{i \in Q^\star(m)} x_i^{m,t} \leq 1, \qquad m = 1, \ldots, M,\ t = 1, \ldots, T \tag{3.2}$$

$$\sum_{m=1}^{M} \sum_{i \in Q(m)} \sum_{t=1}^{T-E^i+1} P_i(c,k)\, x_i^{m,t} \geq d(c,k), \qquad c = 1, \ldots, C,\ k = 1, \ldots, q_c \tag{3.3}$$

$$(E_i - 1)\, x_i^{m,t} \leq \sum_{\alpha=1}^{E_i-1} x_0^{m,t+\alpha}, \qquad \begin{array}{l} m = 1, \ldots, M, \\ t = 1, \ldots, T - E_i + 1, \\ i \in Q(m) \end{array} \tag{3.4}$$

$$x_0^{m,1} = 0 \qquad m = 1, \ldots, M, \tag{3.5}$$

$$x_0^{m,t} \leq \sum_{\gamma=2}^{R} \sum_{j=\gamma}^{R} \sum_{i \in \{Q(m) \cap S_j\}} x_i^{m,t-\gamma+1}, \qquad m = 1, \ldots, M,\ t = 2, \ldots, T \tag{3.6}$$

$$M\, z_t \geq \sum_{m=1}^{M} \left(\sum_{i \in Q^*(m)} x_i^{m,t} \right), \qquad t = 1, \ldots, T \tag{3.7}$$

$$\sum_{i \in Q^*(m)} x_i^{m,t} \geq \sum_{i \in Q^*(m)} x_i^{m,t+1}, \qquad m = 1, \ldots, M,\ t = 1, \ldots, T-1 \tag{3.8}$$

$$\sum_{h\in H_w} y_{h,w} + \sum_{\mu\in\mathbb{O}} a_{w,\mu} o_\mu \leq e_w, \qquad w = W+1,\ldots,W+V \tag{3.9}$$

$$\sum_{h\in H_w} y_{h,w} + \sum_{v=1}^{V} \sum_{h\in H_w(v)} y_{h,w,v} \leq e_w, \qquad w = 1,\ldots,W \tag{3.10}$$

$$\sum_{w=1}^{W+V} \sum_{h\in H_w} a_{\gamma,h,w} y_{h,w} + \sum_{w=1}^{W} \sum_{v=1}^{V} \sum_{h\in H_w(v)} a_{\gamma,h,w,v} y_{h,w,v} + \sum_{\mu\in\mathbb{O}(\gamma)} o_\mu = \sum_{m\in G(\mathcal{L}_\gamma)} \sum_{t=1}^{T} \sum_{i\in Q(m)} D_{c_i} x_i^{m,t}, \qquad \gamma = 1,\ldots,\Gamma \tag{3.11}$$

$$x_i^{m,t} \in \{0,1\}, \qquad m = 1,\ldots,M,\ t = 1,\ldots,T,\ i \in Q^*(m) \tag{3.12}$$

$$z_t \in \{0,1\}, \qquad t = 1,\ldots,T \tag{3.13}$$

$$y_{h,w} \in \mathbb{Z}_+, \qquad w = 1,\ldots,W,\ h \in H_w \tag{3.14}$$

$$y_{h,w,v} \in \mathbb{Z}_+, \qquad w = 1,\ldots,W,\ v = 1,\ldots,V,\ h \in H_w(v) \tag{3.15}$$

$$o_\mu \in \mathbb{Z}_+, \qquad \mu \in \mathbb{O}. \tag{3.16}$$

The objective function (3.1) is divided into 4 terms. The first term is the makespan value. The second term defines the waste related to the use of new bars to produce the demand of bars. The third term describes the waste associated to the use of new bars to produce the bars required by beam production while creating new leftovers. Finally, the fourth term specifies the waste corresponding to the bar leftovers in stock that are used to produce the amount of bars required. Note that each term of (3.1) could alternatively be regarded as an independent objective function to be minimized. We obtain (3.1) using the weighted sum method, in which the parameters $\lambda_i \in \mathbb{R}_+$, with $i = 1,\ldots,4$, indicate the weight of each objective function term. A solution that minimizes (3.1) is, therefore, a Pareto optimum [12].

Constraints (3.2) ensure that at most one pattern must be assigned to mold m at period t, with the possibility of this pattern being P_0. Constraint set (3.3) requires that all demands must be satisfied. Constraints (3.4) force that, if pattern P_i is initiated at period t, then the next $E_i - 1$ periods shall have the pattern P_0 assigned to them (the right-hand side of the constraint remains unconstrained, in case $x_i^{m,t} = 0$). Constraint sets (3.5) and (3.6) establish that P_0 shall only be used in mold m if there is some pattern associated with a previous period in the same mold, whose production has not yet been completed.

Each constraint in set (3.7) ensures that variable z_t must be 1 if period t is used to produce beams. Constraints (3.8) force that there is no inactive period during beam production in the molds. This means that the production is continuous, *i.e.*, if a mold is used it will be used with no interruption; in other words, if the production stops at a given mold and period, it will not resume in that mold at a subsequent period.

Constraints (3.9) establish that the number of bar leftovers cut plus the number of leftover bars used to produced bars *via* overlapping does not exceed the stock, note that the cutting of a leftover does not generate leftovers. Constraint set (3.10) ensures that the number of bars cut does not exceed the stock. Constraints (3.11) force that the amount of bars necessary to produce the beams is achieved, assuming that the required amount of bars is the number of bars used by the forms in the entire time horizon. Constraints (3.12)–(3.16) define the domains of the decision variables.

The model (ICP) has $\mathcal{O}(MTr + WVH + O)$ variables and $\mathcal{O}(q + MTr + V + W + \Gamma)$ constraints, with $q = \sum_{i=1}^{C} q_c$. Thus, depending on the total number of possible packing, cutting, and overlapping patterns, there may be an excessive number of variables and constraints in the model. We choose to limit the number of packing patterns, which are the more numerous type of pattern, in practice, by using only maximal packing

patterns, used successfully by [1,21]. We say that a pattern P_i *contains* a pattern P_j if $c_i = c_j$ and $a_k^i \geq a_k^j$, with $k = 1, \ldots, q_{c_i}$.

Proposition 3.1. *Restricting the model (ICP) to using only maximal packing patterns does not modify its set of optimal solutions.*

Proof. The proof can be found in [1]. □

3.2. NP-hardness

To argue the ICP-HPBMPP hardness note that for instances where $D_c = 0$, for all $c = 1, \ldots, C$, constraints (3.9)–(3.11) are naturally fulfilled and all variables $y_{h,w}, y_{h,w,v}$ and o_μ are set to zero, reducing an instance of ICP-HPBMPP to an HPPMBPP instance involving the minimization of the makespan, up to a constant multiplicative factor. Consequently, the ICP-HPBMPP is a generalization of HPPMBPP, which is already known to be NP-hard [1].

3.3. Objective function lower bound

Since the ICP-HPBMPP is a NP-hard problem, a lower bound for the optimal objective function value may help in evaluating the quality of feasible solutions in heuristic and exact methods. In order to simplify the presentation of our proposed lower bound for objective function (3.1) optimal value, we present the following notation. For a given $\gamma \in \{1, \ldots, \Gamma\}$ we define the following sets:

- $C1_\gamma = \{f_{h,w}/a_{\gamma,h,w} : h \in H_w \quad \wedge \quad a_{\gamma,h,w} > 0 \quad \wedge \quad 1 \leq w \leq W\}$.
- $C2_\gamma = \{\alpha' f_{h,w,v}/a_{\gamma,h,w,v} : h \in H_w(v) \quad \wedge \quad a_{\gamma,h,w,v} > 0 \quad \wedge \quad 1 \leq w \leq W \quad \wedge \quad 1 \leq v \leq V\}$.
- $C3_\gamma = \{\alpha'' f_{h,w}/a_{\gamma,h,w} : h \in H_w \quad \wedge \quad a_{\gamma,h,w} > 0 \quad \wedge \quad W+1 \leq w \leq W+V\}$.
- $C4_\gamma = \{\alpha'' f_\mu : \mu \in \mathbb{O}(\gamma)\}$.
- $\hat{C}_\gamma = C1_\gamma \cup C2_\gamma \cup C3_\gamma \cup C4_\gamma$.

An upper bound on the optimal value of model (ICP) is given by equation (3.17).

$$\left\lceil \sum_{c=1}^{C} t_c \cdot \left(\sum_{k=1}^{q_c} l(c,k) \cdot d(c,k) \right) \Big/ \sum_{m=1}^{M} L_m \right\rceil + \min_{\gamma \in \{1,\ldots,\Gamma\}} \left\{ \left\lceil \sum_{c=1}^{C} D_c \cdot \left(\sum_{k=1}^{q_c} l(c,k) \cdot d(c,k) \right) \Big/ \mathcal{L}_\gamma \right\rceil \cdot \min\{\hat{C}_\gamma\} \right\}. \tag{3.17}$$

The first part of equation (3.17) corresponds to a lower bound for the makespan, while the second part stands for the minimum waste resulting from using molds of some fixed length $\mathcal{L}_\gamma$.

4. Patterns generation

Instead of carrying out exhaustive enumerations, we generated the desirable packing, cutting, and overlapping patterns for a given instance using constraint programming models, which are described in the remainder of this section.

4.1. Packing patterns generation

Consider the following notation, in addition to the notation presented in Section 3:

- K: the largest number of different lengths among beam types, *i.e.*, $\max q_c$ with $c = 1, \ldots, C$. For example, in an instance with 2 beam types, in which type 1 has 6 distinct beam lengths and type 2 has 4 distinct beam lengths, we have $K = 6$.
- $v_i \in \{1, \ldots, C\}$: a decision variable that corresponds to the type of beam used by the pattern P_i.
- $\gamma_i \in \{1, \ldots, \Gamma\}$: auxiliary decision variable for generating patterns that will be maximal in at least one mold of the problem. It defines in which mold capacity the generated pattern P_i is maximal.

- $A^i \in \mathbb{Z}^K$: a vector of decision variables, with A_j representing the number of beams of the length $\ell(v,j)$, for all $j \in \{1,\ldots,K\}$. Given a pattern P_i of type v, the nonzero components of vector A^i correspond to $[\mathcal{N}_i(v,j)]_{j=1}^{q_v}$.
- $P_i = \left(v_i, (A_1^i, \ldots, A_{q_v}^i)\right)$: the generated pattern.

For the generation of a packing pattern P_i we propose the model, which is adapted from [1].

$$1 \leq v_i \leq C, \tag{4.1}$$

$$1 \leq \gamma_i \leq \Gamma, \tag{4.2}$$

$$A_j^i = 0, \text{ if } v_i = c, \qquad c = 1,\ldots,C, \quad j = q_c + 1,\ldots,K \tag{4.3}$$

$$\mathcal{L}_m - \min_{j=1,\ldots,q_c}(l(c,j)) < \sum_{j=1}^{q_c} l(c,j)\cdot A_j^i \leq \mathcal{L}_m, \text{ if } (v_i = c \wedge \gamma_i = m), \qquad c = 1,\ldots,C, \quad m = 1,\ldots,\Gamma, \tag{4.4}$$

$$A_k^i \in \mathbb{Z}_+, \qquad k = 1,\ldots,K. \tag{4.5}$$

Constraint (4.1) implies that the pattern type has domain $\in \{1,\ldots,C\}$. Constraint (4.2) defines the length of the molds in which the generated pattern should be maximal. Constraint set (4.3) implies that if the generated pattern is of type v then it includes no beam of size $l(v,j)$, such that $j > q_v$. Constraint set (4.4) imposes that the capacity used by the generated pattern is simultaneously larger than the mold length minus the shortest beam length from its type and no larger than the length of the actual mold. The empty pattern is, therefore, not generated and has to be manually included in the final set of patterns. We utilized the solver CPLEX CP Optimizer to enumerate all the solutions of model (4.1)–(4.5).

4.2. Cutting patterns generation

In this section, we propose a constraint programming model for cutting patterns generation. The decision variables are given below:

- w_h: index of the bar that will be cut in the generated cutting pattern I_h;
- A_i^h: number of items of length $\mathcal{L}_i$ cut in the pattern, for $i \in \{1,\ldots,\Gamma\}$;
- A_i^h: number of items of length b_{W+i} cut in the pattern, for $i \in \{\Gamma+1,\ldots,\Gamma+V\}$;
- $I_h = \left(w_h, \left(A_1^h,\ldots,A_\Gamma^h, A_{\Gamma+1}^h,\ldots,A_{\Gamma+V}^h\right)\right)$: the generated pattern.

The proposed constraint model for generating a cutting pattern H_h is given by equations (4.6)–(4.10).

$$1 \leq w_h \leq W + V, \tag{4.6}$$

$$\sum_{i=1}^{\Gamma} \mathcal{L}_i \cdot A_i^h + \sum_{i=1}^{V} b_{W+i} \cdot A_{\Gamma+i}^h \leq \text{element}(w_h, b), \tag{4.7}$$

$$\#\{i \in \{\Gamma+1,\ldots,\Gamma+V\} | A_i^h > 0\} = 1, \tag{4.8}$$

$$A_i^h = 0, \text{ if } w_h > W, \qquad i = \Gamma+1,\ldots,\Gamma+V \tag{4.9}$$

$$A_i^h \in \mathbb{Z}_+, \qquad i = 1,\ldots,\Gamma+V. \tag{4.10}$$

Constraint (4.6) defines the domain of each decision variables w_h. Each w_h variable determines defines the bar that will be cut in the current pattern to generate items. If $1 \leq w \leq W$, the bar that will be cut is a new bar. If $W+1 \leq w \leq W+V$, the bar that will be cut is a bar leftover. Constraint (4.7) states that the total length of items cut in the pattern must be shorter than the length of the bar used to cut such pattern, with expression element(w_h, b) standing for the w_h-th element of array b [4]. Constraint set (4.8) implies that a cutting pattern only generates one type of leftover. Constraint (4.9) implies that a leftover does not generate more leftovers. We utilized the CPLEX CP Optimizer to enumerate all the solutions of model (4.6)–(4.10).

4.3. Overlapping patterns

In order to enrich the problem by allowing the possibility of using overlapping bars, we recall that an overlapping pattern $\mathbb{O}_\mu$ is a tuple $\mathbb{O}_\mu = (\gamma_\mu, (a_1^\mu, \ldots, a_V^\mu))$. Note that γ is associated to the length of the bar that is generated in such pattern. Such length must be equal to the capacity of some mold, since we are only required to produce bars *via* overlapping that are used for beam production. A bar produced by overlapping is only produced from leftovers in stock.

In order to simplify the model's notation, consider the following decision variables:

- A_i^μ: decision variable that represents the number of items b_{W+i} used in the overlapping pattern, for $i \in \{1, \ldots, V\}$.
- $\gamma_\mu \in \{1, \ldots, \Gamma\}$: decision variable that defines the length of the bar produced by the overlapping pattern.
- $f \geq 0$: decision variable that expresses the waste of bar associated to the overlapping pattern to produce a bar of length $\mathcal{L}_\gamma$.
- $\mathbb{O}_\mu = (\gamma, (A_1^\mu, \ldots, A_V^\mu))$: the generated pattern.

The following constraint programming model can be used to produce an overlapping pattern:

$$1 \leq \gamma_\mu \leq \Gamma, \tag{4.11}$$

$$\sum_{i=1}^{V} A_i^\mu b_{W+i} \geq \mathcal{L}_{\gamma_\mu} + \epsilon, \tag{4.12}$$

$$\sum_{i=1}^{V} A_i^\mu = 2, \tag{4.13}$$

$$f = \mathcal{L}_{\gamma_\mu} - \sum_{i=1}^{V} A_i^\mu b_{W+i}. \tag{4.14}$$

Constraint (4.11) ensures that the length of the bar produced is one of the possible mold lengths. Constraint (4.12) forces that the total length of the chosen leftovers is greater than the length of the bar produced *via* overlapping plus a constant ϵ which is the loss of the bar resulting from the overlapping process. Constraint (4.13) defines that only 2 leftovers are used in the production of the bar made *via* overlapping. Constraint (4.14) defines the bar waste resulting from the overlapping pattern.

The constraint programming model for overlapping pattern generation is sufficiently flexible to accommodate the production planner's necessities. In a more general setting, we could require that a bar made *via* overlapping can only be produced by using more than 2 and no more than a predefined number of leftovers and specify the ϵ value to be proportional to the number of leftovers used in such pattern.

5. Genetic algorithm for the ICP-HPBMPP

In this section, we propose a genetic algorithm to solve the ICP-HPBMPP, formalize the solution representation chosen, the solution fixing procedure, the selection, mutation, and crossover operators, as well as the initial population generation, population restart, and local search.

5.1. Solution representation

The solution representation consists of a 2-row matrix, in which each column j consists of the genes a_j and x_j, where a_j is a pattern index and x_j is the number of times the pattern represented by a_j is used. The number of columns of this representation is variable and can be at most $r + H + O$. The a_j genes can have values in $\{1, \ldots, r+H+O\}$, in which the values $1, \ldots, r$ represent the packing patterns indices, the values $r+1, \ldots, r+H$ correspond to the cutting patterns indices, and the values $r + H + 1, \ldots, r + H + O$ are associated with the indices of overlapping patterns. In Figure 2, we show a generic scheme of the solution representation in which the number of columns is exactly $r + H + O$.

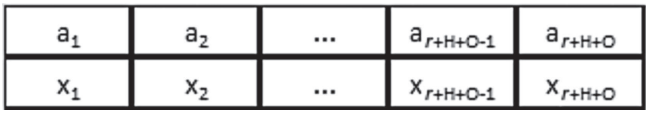

FIGURE 2. Solution representation.

TABLE 1. Description of instance cwp000.

Instance cwp000
$C = 1 \quad M = 5 \quad T = 3$
$W = 1 \quad V = 4$
$L = (5.95, 5.95, 5.95, 5.95, 11.95)$
$t_1 = 1$
$q_1 = 2$
$D_1 = 1$
$l(1, \cdot) = (1.12, 3.3)$
$d(1, \cdot) = (5, 10)$
$b = (12, 2, 5, 6, 8)$
$e = (30, 16, 28, 25, 29)$
$\epsilon = 0.3$

TABLE 2. Packing patterns for instance cwp000.

ID	Beam type	Capacity	a_1^p	a_2^p
1	1	5.6	5	0
2	1	5.54	2	1
3	1	11.2	10	0
4	1	11.14	7	1
5	1	11.08	4	2
6	1	11.02	1	3

In order to illustrate the solution representation we first present, in Table 1, the cwp000 instance, which was randomly generated. Its respective packing, cutting, and overlapping patterns are presented in Tables 2, 3, and 4, respectively.

Note that ID is associated with the pattern indices. An optimal solution for the cwp000 instance is shown as the chromosome in Figure 3.

In the solution shown in Figure 3, we obtain an objective function value of 2.1, with makespan of 2 periods and bar waste of 0.1 m. Figure 4 shows that packing patterns with indices 2 and 6 were used 4 and 2 times, respectively. Due to the fact that we are restricted to using only maximal packing patterns in their respective molds and a given packing pattern is maximal with respect to only one particular length of mold, we infer that packing pattern 2 is associated with molds of length 5.95 m, and packing pattern 6 is associated with molds of length 11.95 m. Therefore, we need to produce a total number of 2 bars of length 5.95 m and 6 bars of length 11.95, since the beam type produced by each solution packing pattern requires only one bar. The cutting patterns used are those with indices 11, 15 and 16, and their frequencies are 2, 1, and 2, respectively. None of the overlapping patterns was selected in the solution.

TABLE 3. Cutting patterns for instance cwp000.

ID	Bar cut	Capacity	a_1^h	a_2^h	a_3^h	a_4^h	a_5^h	a_6^h
7	1	5.95	1	0	0	0	0	0
8	1	7.95	1	0	1	0	0	0
9	1	9.95	1	0	2	0	0	0
10	1	11.95	1	0	3	0	0	0
11	4	5.95	1	0	0	0	0	0
12	5	5.95	1	0	0	0	0	0
13	1	11.95	1	0	0	0	1	0
14	1	10.95	1	0	0	1	0	0
15	1	11.9	2	0	0	0	0	0
16	1	11.95	0	1	0	0	0	0

TABLE 4. Overlapping patterns for instance cwp000.

ID	Bar generated	Waste of bar	a_1^μ	a_2^μ	a_3^μ	a_4^μ
17	1	1.05	1	1	0	0
18	1	4.05	0	2	0	0
19	1	2.05	1	0	1	0
20	1	6.05	0	0	2	0
21	1	5.05	0	1	1	0
22	1	8.05	0	0	1	1
23	1	4.05	1	0	0	1
24	1	7.05	0	1	0	1
25	1	10.05	0	0	0	2
26	2	4.05	0	0	0	2
27	2	1.05	0	1	0	1
28	2	2.05	0	0	1	1

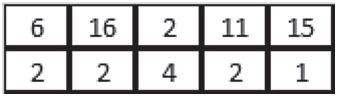

FIGURE 3. Example of a feasible solution for instance cwp000.

The production planning consists of the specification of the exact quantity of bars required for the beam production, as long as the available stock of bars is not violated. Thus, the solution encoded in the chromosome in Figure 3 is feasible.

5.2. Initial population generation

Since we typically need a large quantity of individuals to generate a population, deterministic methods are not the best choice, despite the high-quality solutions produced by them. We propose a pseudorandom approach to generate a large quantity of solutions, which is described in Algorithm 1.

We call this method pseudorandom because we choose the patterns to add to the solution randomly, although each pattern frequency in the solution is computed in such a way as to respect stock and satisfy the demand.

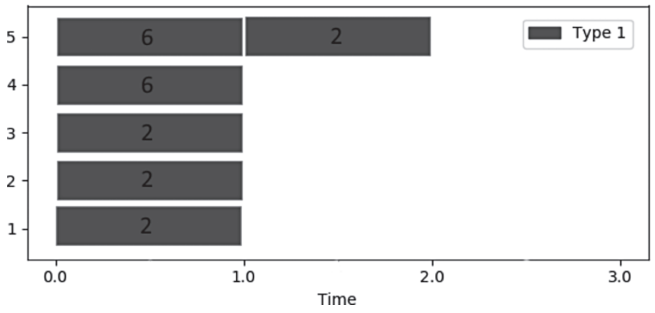

FIGURE 4. Gantt chart for an optimal solution of instance cwp000.

Algorithm 1: Generate pseudo-random solution.

Data: Instance, Set of Packing Patterns, Set of Cutting Patterns, Set of Overlapping Patterns
Result: Feasible solution
Initialize *solution* with all patterns with their respective frequencies set to zero.
while *Beam demands is not fulfilled* **do**
  $pac_p \leftarrow$ random packing pattern that has not yet been selected.
  if *There is some beam in* pac_p *whose demand is unfulfilled* **then**
    Increment the number of times that pac_p is used in *solution* until all beams in pac_p have their demands fulfilled.
  end
end
Calculate the number of bars needed according to the packing patterns frequencies
for *each mold length* γ **do**
  while *(number of bars of length* $\mathcal{L}_\gamma$ *needed was not reached)* $\vee$ *(there is at least one cutting pattern not selected)* **do**
    $cut_p \leftarrow$ random cutting pattern that generates bars of length $\mathcal{L}_\gamma$ that has yet not been selected.
    bars_needed $\leftarrow$ number of bars of length $\mathcal{L}_\gamma$ required.
    $n \leftarrow$ number of times cut_p can be added to *solution* without violating bars stock.
    Increment cut_p frequency in *solution* by $max(bars\_needed, n)$ times.
  end
  while *number of bars of length* $\mathcal{L}_\gamma$ *needed was not reached* **do**
    $ove_p \leftarrow$ random overlapping pattern that generates a bar of length $\mathcal{L}_\gamma$ that has not yet been selected.
    bars_needed $\leftarrow$ number of bars of length $\mathcal{L}_\gamma$ required.
    $n \leftarrow$ number of times ove_p can be added to *solution* without violating bars stock.
    Increment ove_p frequency in *solution* by $max(bars\_needed, n)$ times.
  end
end
Remove from *solution* the genes associated to patterns that are not used
return *solution*

The time complexity of the Algorithm 1 is $\mathcal{O}(Pq_c + \Gamma(H + O))$. Generating the initial population consists of creating of a number of individuals with the use of Algorithm 1 and selecting the best of them based on their fitness value according to the required population size.

5.3. Fitness function and selection operator

We use the objective function (3.1) from the mathematical model (ICP) as the fitness function to evaluate the solution quality of a given chromosome. The selection operator consists of the process of selecting the best distinct solutions with respect to their respective fitness function value, *i.e.*, the individuals with the lowest fitness values.

5.4. Crossover operators

In this subsection, we propose two alternatives to use as crossover operators: types 1 and 2. Given two parents, both crossover types generate one offspring, which consists of a new solution (chromosome).

In crossover type 1, we preserve all pattern indices from both parents, but the number of times each pattern is used in the offspring corresponds to the mean of the number of times they are used by the parents rounded to the largest integer. For each gene there is a probability of mutation. When the mutation occurs the number of times that the current pattern is used in such gene is set to zero. After this crossover process, if the generated offspring results in an infeasible solution, an iterative procedure, shown in Algorithm 6, is applied for its correction. If some pattern from the current offspring is used zero times, the gene associated to it is removed from the chromosome.

In crossover type 2, we first initiate the offspring using all patterns that used in both parents with their respective frequencies set to zero. For the genes that have patterns that are part of both parents simultaneously, their respective frequencies are set as the mean of their frequencies in the parents rounded to the largest integer. For each remaining gene we have a probability of 50% of setting its respective frequency to be equal to the originating parent frequency or keeping it equal to zero. If the resulting offspring is not feasible, the fixing procedure, shown in Algorithm 6, is applied to it and all patterns with final frequencies equal to zero have their respective genes removed from the chromosome.

5.5. Mutation operator

The mutation of an individual consists of (a) choosing one pattern p_1 that is in the solution, and (b) adding one pattern p_2, chosen randomly, that is not part of the solution. The number of times that p_2 is used becomes the number of times that p_1 is used, and the number of times that p_1 is used is set to zero. If the solution is infeasible after this procedure we apply a fixing phase to the solution. This process is frequently required in practice and is described in the next subsection.

5.6. Infeasible solution fixing

Since the proposed genetic operators of crossover and mutation can compromise the feasibility of solutions, we must define a procedure to fix infeasible solutions, turning them into feasible ones.

A chromosome may be an infeasible solution due to different reasons, as follows:

(1) Infeasibility type 1, due to beam demand: the frequencies of packing patterns in the solution are not enough to fulfill the beam demands;
(2) Infeasibility type 2, due to bar stock: the number of bars which are used in cutting and overlapping patterns exceeds the bar stock;
(3) Infeasibility type 3, due to inconsistent number of bars produced and required: the number of necessary bars generated by cutting and overlapping patterns is different from the number of bars that beam production requires.

If we detect any of those kinds of infeasibility, we must apply the infeasible solution fixing phase, which consists of Algorithm 6. Each infeasibility type is treated in a particular procedure: Algorithms 3, 4, and 5 are used to fix infeasibility types 1, 2, and 3, respectively.

The unnecessary packing patterns procedure, shown in Algorithm 2, works like a solution treatment phase, which is not a necessary part of the solution fixing process, although applying such procedure may improve solution quality and simplify the fixing process, *i.e.*, it would be less likely that the modified solutions could not be fixed. The procedure consists of decreasing the frequency of packing patterns after the beam demands are already fulfilled if there are beam surplus.

In Figure 5, we show an example of the crossover operators, with offspring 1 as the solution generated by crossover operator type 1, and offspring 2 as the solution created by crossover operator type 2. Note that the fixing procedure was applied to offspring 2 and not to offspring 1. In Figure 6, we show an example of the

Algorithm 2: Remove unnecessary packing patterns.

```
Data: Infeasible chromosome
Result: Potentially modified chromosome
Initialize produced beams with zeros;
demand_fulfilled ← false;
for each packing pattern P_i in Chromosome do
    if demand_fulfilled = false then
        for cont = 1,..., frequency(P_i) do
            Update produced beams;
            if produced beams fulfill the beam demands then
                demand_fulfilled ← true;
                frequency(P_i) ← cont;
                break;
            end
        end
    else
        frequency(P_i) ← 0
    end
end
return Chromosome
```

Algorithm 3: Fix chromosome with respect to infeasibility type 1.

```
Data: Infeasible chromosome
Result: Potentially feasible chromosome
while Infeasibility type 1 = true do
    for each beam type c do
        for each beam length l_c whose demand is not fulfilled do
            for each packing pattern P_i with type c in Chromosome do
                if frequency of l_c in P_i ¿ 0 then
                    Increment frequency(P_i) until the demand of l_c is achieved;
                    break;
                end
            end
        end
    end
end
return Chromosome
```

proposed mutation operator. The resulting chromosome is infeasible; therefore, the solution fixing procedure must be applied. If the application of the solution fixing procedure to a given chromosome could not turn it into a feasible solution, the chromosome is discarded.

5.7. Population restart

The population restart consists of the creation of a new population to compose the next generation after a predefined number of epochs. We apply a population restart after a given number of generations with no improvement of the best-fitness value. We divide such procedure into three parts, as follows: (1) selecting a certain number of the best-fitness individuals from the current population; (2) generating a number new pseudo-random individuals; (3) creating a new population with individuals from steps 1 and 2 and applying the selection operator to form the next population.

Algorithm 4: Fix chromosome with respect to infeasibility type 2.

Data: Infeasible chromosome
Result: Potentially feasible chromosome
Calculate the #bars used;
for *each standard bar or bar leftover* w **do**
  if *#bars* w *used ¿ stock of* w *bars* **then**
    for *each cutting pattern* I_h *that uses* w *in Chromosome* **do**
      rt $\leftarrow$ #bars w used − stock of w bars;
      frequency(I_h) $\leftarrow$ frequency(I_h) − min(frequency(I_h), rt);
      Update the #bars w used;
      if *#bars* w *used > stock of* w *bars* **then**
        break;
      end
    end
    for *each overlapping pattern* O_μ *that uses* w *in Chromosome* **do**
      rt $\leftarrow$ #bars w used − stock of w bars;
      rt $\leftarrow \left\lfloor \frac{\text{rt}}{\#\text{bars } w \text{ in } O_\mu} \right\rfloor$
      frequency(O_μ) $\leftarrow$ frequency(O_μ) − min(frequency(O_μ),rt);
      Update the #bars w used;
      if *#bars* w *used > stock of* w *bars* **then**
        break;
      end
    end
  end
end
return *Chromosome*

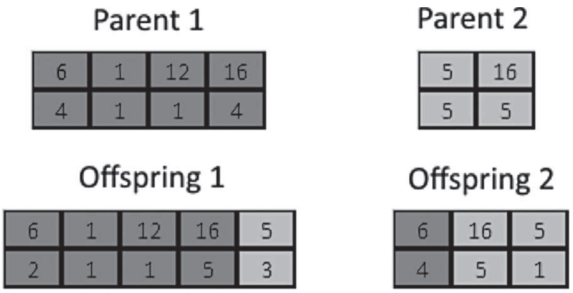

FIGURE 5. Crossover operators.

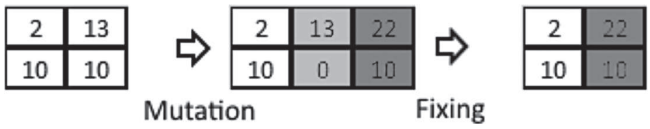

FIGURE 6. Mutation operator and solution correction.

5.8. Local search

In order to improve the quality of final solutions, we apply a local search to every individual of the final population. For the local search we use the *insert* movement, which consists of, given two genes indices i and k, with $i < k$, inserting the gene i one position in front of k-th gene, *i.e.*, all the genes between positions i and

Algorithm 5: Fix chromosome with respect to infeasibility type 3.

Data: Infeasible chromosome
Result: Potentially feasible chromosome
Calculate the #bars generated by cutting and overlapping patterns;
Calculate the #bars that beam production requires according to the frequency of packing patterns;
for *each bar* γ *generated* **do**
  if *#bars* γ *generated ¿ #bars* γ *that beam production requires* **then**
    for *each cutting pattern* I_h *that generates only bars* γ **do**
      rt $\leftarrow$ #bars γ generated $-$ #bars γ that beam production requires;
      rt $\leftarrow \left\lceil \frac{\text{rt}}{\#\text{bars } \gamma \text{ generated by } I_h} \right\rceil$
      frequency(I_h) $\leftarrow$ frequency(I_h) $-$ min(frequency(I_h), rt);
      Update the #bars γ generated;
    end
  end
  if *#bars* γ *generated* $>$ *#bars* γ *that beam production requires* **then**
    for *each overlapping pattern* O_μ *that generates a bar* γ **do**
      rt $\leftarrow$ #bars γ generated $-$ #bars γ that beam production requires;
      frequency(O_μ) $\leftarrow$ frequency(O_μ) $-$ min(frequency(O_μ), rt);
      Update the #bars γ generated;
    end
  end
  if *#bars* γ *generated* $<$ *#bars* γ *that beam production requires* **then**
    for *each cutting pattern* I_h *that generates only bars* γ **do**
      rt $\leftarrow$ #bars γ that beam production requires $-$ number bars γ generated;
      rt $\leftarrow \left\lfloor \frac{\text{rt}}{\#\text{bars } \gamma \text{ generated by } I_h} \right\rfloor$
      frequency(I_h) $\leftarrow$ frequency(I_h) + min(rt, stock of γ bars remaining) ;
      Update the #bars γ generated;
    end
  end
  if *#bars* γ *generated* $<$ *#bars* γ *that beam production requires* **then**
    for *each overlapping pattern* O_μ *that generates a bar* γ **do**
      Increment frequency(O_μ) until (#bars γ generated $\geq$ #bars γ that beam production requires) or the stock is violated with new increment;
      Update the #bars γ generated;
    end
  end
end
return *Chromosome*

$k+1$ are moved one position to the right after the insertion of the k-th gene. In Figure 7 an insert movement neighbor is shown for a given solution after inserting 2nd gene in front of 5th gene.

Considering the function INSERT(solution, i, k) as the movement of insertion given indices i and k, we describe the local search procedure in the Algorithm 7.

5.9. Algorithm description

In order to describe the proposed genetic algorithm we define the following parameters: population size (TP), number of generations (NG), crossover type (CRS), number of pseudo-random solutions generated for the initial population and restart selections (AS), mutation probability (MUT), number of generations with

Algorithm 6: Solution fixing procedure.

Data: Infeasible chromosome
Result: Possible feasible chromosome
if *Infeasibility type 1 = true* **then**
  Call Algorithm 3;
else
  Call Algorithm 2;
end
if *Infeasibility type 2 = true* **then**
  Call Algorithm 4;
end
if *Infeasibility type 3 = true* **then**
  Call Algorithm 5;
end
return *chromosome*

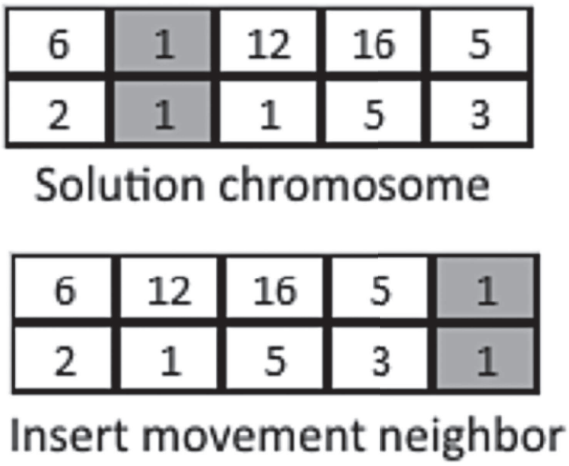

FIGURE 7. Insert movement.

Algorithm 7: Insert neighborhood.

Data: *InitialSolution*
Result: *BestSolution*
$BestSolution \leftarrow InitialSolution$;
for $i = 1, \ldots, n\ell - 1$ **do**
  for $k = k + 1, \ldots, n\ell$ **do**
    $neighbor \leftarrow \text{INSERT}(InitialSolution, i, k)$;
    if $makespan(neighbor) < makespan(BestSolution)$ **then**
      $BestSolution \leftarrow neighbor$;
    end
  end
end
return *BestSolution;*

no fitness improvement to apply population restart (RST), and the number of individuals from the current population selected to be used in restart operator procedure (TER).

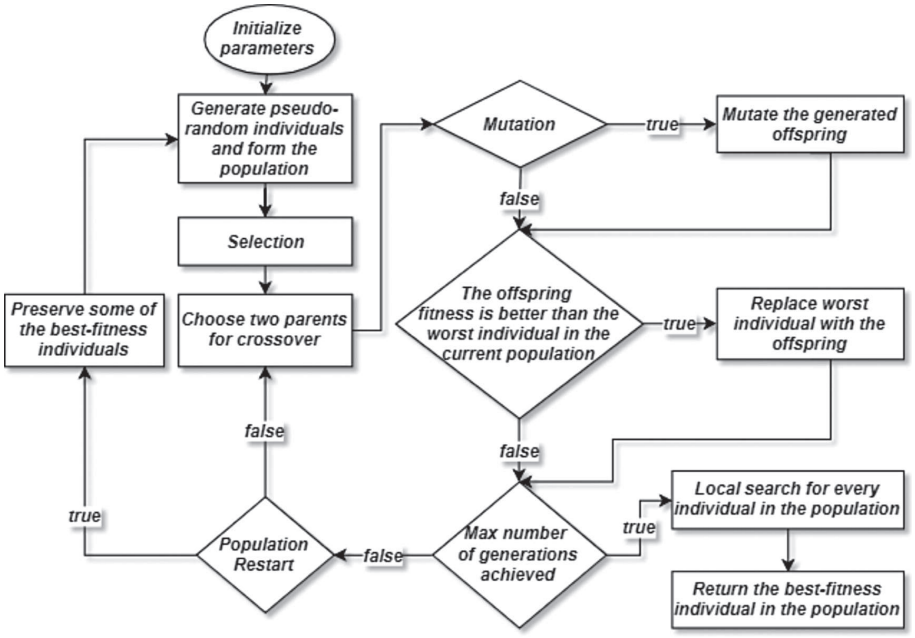

FIGURE 8. Simplified flowchart of proposed genetic algorithm.

The proposed genetic algorithm can be seen as a steady-state model since only one new individual is created per generation, even though we create several individuals in the formation of the initial population and in a population restart process. A simplified scheme of the proposed genetic algorithm is shown in the flowchart in Figure 8.

6. Computational experiments

In this section, we present computational experiments on a set of benchmark instances that were produced with the intent to mimic real-world scenarios, in order to evaluate the solution methods proposed in this study. The instances used in this work were randomly generated based on an existing order arising from a real-world production plant [1, 3]. For privacy reasons, we are not allowed to use or provide here the actual data coming from the aforementioned instance.

The patterns corresponding to each test instance were generated using the constraint programming solver IBM ILOG CPLEX 12.8 CP Optimizer. For the integer programming model implementation we adopted the solver IBM ILOG CPLEX 12.8. Both solvers were used with Concert technology using the C++ programming language. The genetic algorithms were also developed with the C++ programming language.

We carried out every test in this paper on a Linux Ubuntu 18.04 64 bits machine with 8 GB of memory and Intel Core i5-3470 CPU 3.20 GHz ×4 processor. We compiled the created codes with the GNU GCC 7.3.0 compiler using Code::Blocks 17.12 IDE. Note that, for different values of λ_i we can form the Pareto front and may have different behaviors of the proposed model and algorithms. However, for the purpose of the study, we did not approach the multi-objective nature of the problem and considered, for each test described in this section, $\lambda_i = 1$, with $i = 1, \ldots, 4$.

6.1. Test instances generation

In this subsection, we describe how we generate the set of benchmark instances used in this section. We introduce a set of instances that are based on data arising from a possible real-world scenario. The different

TABLE 5. Description of test instances.

Instance	C	M	T	r	H	O
cwp001	1	15	6	145	10	12
cwp002	1	15	6	199	10	12
cwp003	1	15	6	236	10	12
cwp004	1	15	6	210	10	12
cwp005	1	15	6	236	10	12
cwp006	1	30	3	257	10	12
cwp007	1	30	3	257	10	12
cwp008	1	30	3	199	10	12
cwp009	1	30	3	218	10	12
cwp010	1	30	3	199	10	12
cwp011	2	15	15	414	10	12
cwp012	2	15	21	395	10	12
cwp013	2	15	21	361	10	12
cwp014	2	15	14	387	10	12
cwp015	2	15	17	451	10	12
cwp016	2	30	8	466	10	12
cwp017	2	30	8	352	10	12
cwp018	2	30	9	459	10	12
cwp019	2	30	8	500	10	12
cwp020	2	30	9	466	10	12
cwp021	3	15	29	662	10	12
cwp022	3	15	36	643	10	12
cwp023	3	15	30	614	10	12
cwp024	3	15	29	671	10	12
cwp025	3	15	35	684	10	12
cwp026	3	30	15	589	10	12
cwp027	3	30	18	560	10	12
cwp028	3	30	18	433	10	12
cwp029	3	30	17	620	10	12
cwp030	3	30	20	557	10	12
cwp031	4	15	45	952	10	12
cwp032	4	15	50	650	10	12
cwp033	4	15	45	896	10	12
cwp034	4	15	41	839	10	12
cwp035	4	15	41	783	10	12
cwp036	4	30	20	715	10	12
cwp037	4	30	24	679	10	12
cwp038	4	30	15	702	10	12
cwp039	4	30	14	732	10	12
cwp040	4	30	30	750	10	12
cwp041	5	15	68	966	10	12
cwp042	5	15	57	927	10	12
cwp043	5	15	66	985	10	12
cwp044	5	15	59	983	10	12
cwp045	5	15	75	1046	10	12
cwp046	5	30	29	974	10	12
cwp047	5	30	29	926	10	12
cwp048	5	30	24	949	10	12
cwp049	5	30	30	1008	10	12
cwp050	5	30	27	1062	10	12
cwp051	6	15	62	1249	10	12
cwp052	6	15	51	1204	10	12
cwp053	6	15	51	1221	10	12
cwp054	6	15	62	1291	10	12
cwp055	6	15	65	1371	10	12
cwp056	6	30	21	1324	10	12
cwp057	6	30	33	1279	10	12
cwp058	6	30	33	1305	10	12
cwp059	6	30	35	1052	10	12
cwp060	6	30	32	1165	10	12
cwp061	7	15	60	1427	10	12
cwp062	7	15	86	1396	10	12
cwp063	7	15	113	1211	10	12
cwp064	7	15	53	1438	10	12
cwp065	7	15	89	1395	10	12
cwp066	7	30	36	1243	10	12
cwp067	7	30	45	1568	10	12
cwp068	7	30	38	1403	10	12
cwp069	7	30	39	1487	10	12
cwp070	7	30	39	1494	10	12

instances represent a sample of the variability of the problem's parameters, such as number of beam types, number of molds, and mold lengths.

In Table 5, we present details about each test instance parameter. We can see that the number of packing patterns increases as the number of beam types increases. However, the number of cutting and overlapping patterns remains constant because of the fact that we expect that the possible distinct bar lengths are standardized in real-world scenarios and therefore do not lead to variability.

We consider mold capacities of 5.95 m and 11.95 m, while we take 1.12 m, 1.45 m, 2.35 m, 2.5 m, 2.65 m, 2.95 m, and 3.3 m as possible beam lengths. For each instance, the possible curing times may be 1, 2, or 3 periods, chosen randomly when instances have more than 3 beam types. In addition, if the instance has up to 3 beam types, we associate the curing time to the beam type index, for example the beam type 2 needs a curing time of 2 periods. With respect to the number of bars that some beam type demands, we choose randomly a value between 1 and 3 for each beam type. We choose the beam demands uniformly between 17 and 50. For total time horizon T,

we calculate it as the ceiling of 150% of the optimal makespan lower bound, defined in equation (6.1).

$$T = \left\lceil 1.5 \cdot \sum_{i=1}^{C} t_c \cdot \left(\sum_{k=1}^{q_c} l(c,k) \cdot d(c,k) \right) / \sum_{m=1}^{M} L_m \right\rceil . \tag{6.1}$$

For all instances, we consider an unique length of new bars as 12 m and the possible lengths of bar leftovers as 2 m, 5 m, 6 m, and 8 m. We do not vary such lengths along the test instances since, in practice, it is expected that they are standardized. For generating realistic bar stocks we introduce an upper bound for the number of bars needed to fulfill the beam demand as UB (see Eq. (6.2)).

$$\text{UB} = 2 \cdot T \cdot M \cdot \max_{D_c=1,\ldots,C} \{D_c\}. \tag{6.2}$$

We set the stock of new bars of length 12 m equal to UB, whilst we choose the stock of each leftover randomly between $\lceil \text{UB}/5 \rceil$ and UB following an uniform distribution. We implemented the instance generator using the MATLAB programming language.

6.2. Computational experiments with the mathematical model

In this subsection, we discuss the results of the computational tests with the benchmark instance set that we generated following the scheme described in Section 6.1. In Table 6, we show the results of the computational experiments for the model (ICP) and its linear relaxation. The solution time was limited to 3600 s. As regards to notation in Table 6, we consider LB, IP, and LP standing for the optimal objective function value lower bound, best solution value by CPLEX for model (ICP), and its linear relaxation value, respectively. When we say *gap* we mean the relative percentage deviation between the best integer objective and the objective of the best node remaining in the CPLEX *B&C* tree, calculated like this: $gap = 100 \cdot |bestbound - bestinteger|/(1e - 10 + |bestinteger|)$ (0% means a proven optimal solution). We denote by "B&C nodes" the number of nodes generated in the branch-and-cut tree in the solution process, and t(s) as the solution time in seconds.

Table 6 shows that the linear relaxation of all instances could be solved, with the average time of 53.21 s, and with 624.61 s being the longest time to get to the optimal solution. On the other hand, only 11 instances could be solved to optimality by the integer programming model (4 of them solved in the root node of the B&C tree). For 23 instances CPLEX could not even find a single feasible solution, a situation that we denote by "–". Moreover, for 36 instances feasible solutions were found by CPLEX, but the search was not completed within the time limit. The computational test results show that the larger the instance parameter values are, the harder it is to find solutions for the resulting problem. With increased values of the instance parameters, when solutions are found, the optimality gap tends to be worse, *i.e.*, the solutions achieved within the time limit are further from the optimal solution, in terms of objective function value.

We compare the results of the integer linear model (ICP), its linear relaxation, and our lower bound, in equation (3.17), for the optimal value of objective function in the chart in Figure 9.

In Figure 9, the lower bound proposed in this work for the optimal objective function value was greater than the linear relaxation for all test instances and highly close to the objective function values obtained by CPLEX.

6.3. Experimental design and computational experiments with the proposed genetic algorithm

In order to achieve a better parameterization for the robustness of the proposed genetic algorithm, we apply fractional factorial parameter design. The authors of [9] used Taguchi experimental design (see [15]) to achieve improved robustness of the genetic algorithm which they proposed. In this method the optimal parameter choice is found with the analysis of different level combinations of the control factors in an orthogonal array, with no necessity of testing all of the possible level combinations. Table 7 displays the proposed levels for the genetic algorithm parameters (control factors) introduced in Section 5.

TABLE 6. Results of integer programming model and its linear relaxation.

		Mathematical model				Linear relaxation				Mathematical model				Linear relaxation	
Instance	LB	IP	B&C nodes	Gap	t (s)	LP	t (s)	Instance	LB	IP	B&C nodes	Gap	t (s)	LP	t (s)
cwp001	5.55	6.05	981	0.00%	1.2	3.58	0.02	cwp036	22.95	25.80	1394	12.02%	3600.0	15.60	8.39
cwp002	7.60	8.10	0	0.00%	1.2	6.02	0.04	cwp037	33.60	–	–	–	3600.0	22.94	11.47
cwp003	9.25	9.70	189	0.00%	1.8	7.57	0.08	cwp038	24.30	26.30	55 260	0.37%	3600.0	19.86	2.01
cwp004	7.50	8.20	686	0.00%	2.3	5.80	0.04	cwp039	29.05	32.00	79 965	1.32%	3600.0	25.56	3.32
cwp005	8.45	9.70	75	0.00%	2.1	7.38	0.09	cwp040	34.65	–	–	–	3600.0	20.43	30.10
cwp006	8.15	8.15	0	0.00%	1.1	7.51	0.07	cwp041	66.15	–	–	–	3600.0	35.96	148.02
cwp007	3.70	4.15	0	0.00%	1.3	2.76	0.04	cwp042	55.00	–	–	–	3600.0	29.79	51.94
cwp008	5.40	5.50	0	0.00%	0.7	4.55	0.04	cwp043	67.00	–	–	–	3600.0	37.59	99.51
cwp009	6.15	7.20	3 640 484	0.69%	3600.0	5.54	0.08	cwp044	57.75	–	–	–	3600.0	31.42	137.86
cwp010	6.35	7.00	2455	0.00%	3.1	5.83	0.05	cwp045	69.25	–	–	–	3600.0	32.80	118.61
cwp011	17.30	19.70	221 350	0.62%	3600.0	12.31	0.66	cwp046	41.00	–	–	–	3600.0	28.89	26.76
cwp012	22.85	26.85	451 142	0.32%	3600.0	14.74	0.64	cwp047	31.65	35.90	3240	2.09%	3600.0	19.90	19.33
cwp013	23.10	25.50	564 930	0.32%	3600.0	15.43	0.75	cwp048	33.80	38.15	1112	11.12%	3600.0	24.39	29.49
cwp014	13.95	14.90	754 750	0.39%	3600.0	9.75	0.88	cwp049	37.55	47.05	2	13.61%	3600.0	24.11	11.40
cwp015	18.75	21.00	561 739	0.49%	3600.0	12.85	0.89	cwp050	38.40	–	–	–	3600.0	26.84	57.54
cwp016	8.45	8.90	2422	0.00%	49.8	5.84	1.19	cwp051	61.70	77.20	0	17.09%	3600.0	35.92	46.80
cwp017	14.05	15.70	2711	0.00%	33.6	11.55	0.60	cwp052	52.65	–	–	–	3600.0	35.71	56.03
cwp018	15.05	18.80	775 314	1.34%	3600.0	11.53	0.59	cwp053	51.80	57.30	1379	2.87%	3600.0	33.99	20.94
cwp019	13.60	17.50	457 947	0.59%	3600.0	11.04	0.54	cwp054	60.35	–	–	–	3600.0	35.63	139.89
cwp020	14.95	17.40	406 983	1.02%	3600.0	11.40	0.82	cwp055	64.90	–	–	–	3600.0	36.53	45.26
cwp021	30.00	33.00	3554	12.29%	3600.0	18.53	5.71	cwp056	35.20	39.15	1289	4.42%	3600.0	28.82	23.40
cwp022	36.40	39.85	6188	0.98%	3600.0	20.81	5.37	cwp057	43.15	–	–	–	3600.0	28.48	135.68
cwp023	31.50	35.20	6342	5.39%	3600.0	19.36	7.23	cwp058	48.30	–	–	–	3600.0	34.29	71.79
cwp024	25.70	27.45	12 286	0.38%	3600.0	13.33	3.50	cwp059	51.20	–	–	–	3600.0	36.29	54.39
cwp025	34.65	37.40	7495	6.44%	3600.0	19.82	5.04	cwp060	43.30	62.35	0	28.21%	3600.0	29.78	40.41
cwp026	21.20	22.40	12 095	2.11%	3600.0	15.16	5.31	cwp061	69.15	79.80	21	27.35%	3600.0	46.85	336.39
cwp027	22.60	24.15	7226	5.92%	3600.0	15.03	5.03	cwp062	76.40	–	–	–	3600.0	40.02	469.50
cwp028	25.40	26.30	9273	8.06%	3600.0	18.15	6.43	cwp063	108.55	–	–	–	3600.0	53.72	103.78
cwp029	23.50	25.80	7529	5.25%	3600.0	16.67	6.79	cwp064	64.85	73.20	103	2.84%	3600.0	48.86	25.38
cwp030	26.90	30.00	8024	1.09%	3600.0	18.56	4.12	cwp065	86.60	–	–	–	3600.0	45.36	624.61
cwp031	42.80	47.00	1530	9.04%	3600.0	22.88	14.88	cwp066	48.45	–	–	–	3600.0	33.76	187.76
cwp032	50.40	58.10	365	14.42%	3600.0	28.21	23.74	cwp067	60.95	–	–	–	3600.0	40.96	126.30
cwp033	44.20	50.10	3600	12.07%	3600.0	24.26	42.86	cwp068	62.65	82.55	204	35.26%	3600.0	47.89	61.85
cwp034	40.70	44.10	2102	3.91%	3600.0	24.56	6.51	cwp069	51.05	–	–	–	3600.0	34.18	97.62
cwp035	43.35	–	–	–	3600.0	27.33	12.67	cwp070	62.50	–	–	–	3600.0	46.39	137.69

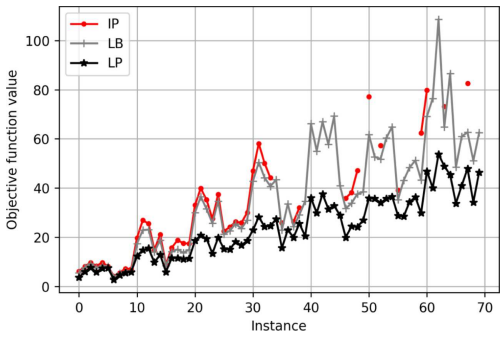

FIGURE 9. Objective function values for integer model solutions, linear relaxation solutions and proposed lower bound value for test instances.

TABLE 7. Factor levels.

Factors	Index of levels	Levels
TP	1	25
	2	50
NG	1	$500{\cdot}r$
	2	$1000{\cdot}r$
MUT	1	0.01
	2	0.025
	3	0.05
RST	1	$\lceil 0.1 \cdot NG \rceil$
	2	$\lceil 0.2 \cdot NG \rceil$
AS	1	$100{\cdot}r$
	2	$500{\cdot}r$
CRS	1	Type 1
	2	Type 2
TER	1	$\lceil 0.1 \cdot Tr \rceil$
	2	$\lceil 0.2 \cdot Tr \rceil$

We must have one degree of freedom for total mean, one degree of freedom for each factor with two levels, and two degrees of freedom for the factor with 3 levels amounting to a total of nine degrees of freedom ($1+1\times 6+2\times 1=9$). However, with the control factors and respective levels that we defined, there is no orthogonal array aside from the full factorial array. Thus, we are not able to use a classical Taguchi orthogonal array design. In such circumstances one alternative is to use a *D-optimal* design approach [6], which aims at minimizing the generalized variance of the estimated regression coefficients. Note that D-optimality is only one possible criterion to choose a particular design. We obtain the *D-optimal* design, by Fedorov algorithm [19] using R programming language for 9 trials for the chosen factors and their respective levels, illustrated in Table 8.

Furthermore, the effectiveness characteristic of the genetic algorithms proposed is the expected fitness value, which we seek to minimize, *i.e.*, "the lower is better principle". Thus, for increased robustness of the algorithm we use the S/N (signal-to-noise) ratio, defined as follows. Note that the larger the value of S/N ratio the better.

$$\text{S/N ratio:} \quad \eta_i = -10 \ln \left(\frac{1}{N} \sum_{j=1}^{N} \text{FIT}_{ij}^2 \right), \tag{6.3}$$

TABLE 8. *D-optimal* design with 9 trials.

Trial	TP	NG	MUT	RST	AS	CRS	TER
1	1	1	1	2	2	1	1
2	1	2	3	1	1	2	1
3	1	2	2	1	2	1	2
4	1	1	2	2	1	2	2
5	2	2	2	2	1	1	1
6	2	1	2	1	2	2	1
7	2	1	3	1	1	1	2
8	2	2	1	1	1	2	2
9	2	2	3	2	2	2	2

with i and j denoting index of trial and index of replication, while FIT stands for the best objective function value obtained by running the GA. We denote by "trial" a certain combination of the control factor levels.

We define a replication as one GA run of some trial for a given instance, and N as the number of test instances multiplied by the number of replications. Since we have an instance set of size 70 and we run each instance 10 times, we perform 700 replications for each trial.

Since CPLEX could not find optimal or even a feasible solution for most instances, we are unable to use the relative percentage deviations from the optimal solution as a performance indicator for the GA. Thus, we use the lower bound relative percentage deviations (LBD) of the fitness values for such purpose. Given a trial i and a replication j the LBD value is defined as follows:

$$\mathrm{LBD}_{ij} = \frac{\mathrm{FIT}_{ij} - \mathrm{LB}_j}{\mathrm{LB}_j}, \tag{6.4}$$

where LB_j stands for the lower bound of the optimal objective function value for the test instance used in replication j. The LBD for a given trial i, denoted by LBD_i, is the average LBD for all replications of instance set, as we can see in the following equation:

$$\mathrm{LBD}_i = \frac{1}{N} \cdot \sum_{j=1}^{N} \mathrm{LBD}_{ij}, \tag{6.5}$$

The remainder of the experimental design procedure consists of three phases:

(1) Evaluate the impacts of the control factors on the S/N ratios and on the LBD values.
(2) For each factor, which has significant impact on the S/N ratios values, we choose the level which increases the S/N ratios.
(3) For the factors, which do not have significant impact on the S/N ratios and have significant impact the LBD values, we choose the level which better approximate the lower bound values.
(4) For the factors, which have significant impact neither on the S/N ratios nor on the LBD values, we select the factor levels regarding the more economic manner, that is, we choose the levels which have less impact on the algorithm running time.

Table 9 shows the results after carrying out the computational tests for each trial with the test instance set. The value of N is 700, which corresponds to 10 replications to each one of the 70 test instances.

In Figure 10, we can see the main effects plot for the control factors using S/N ratios as the response variable. In Figure 11, we show the boxplots for each factor also using S/N ratios as the response variable. The mean response is clearly influenced by the type of crossover, while it is not so clear to affirm whether or not the other factors influence the response variable (Tab. 10).

TABLE 9. LBD, S/N ratio, and average execution time results for each trial.

Trial	Control factors							LBD values	S/N ratios	Average time (s)
	TP	NG	MUT	RST	AS	CRS	TER			
1	1	1	1	2	2	1	1	0.23719	−80.01779	274.7
2	1	2	3	1	1	2	1	0.28382	−80.88757	149.5
3	1	2	2	1	2	1	2	0.21597	−79.52707	589.7
4	1	1	2	2	1	2	2	0.33001	−81.74368	69.0
5	2	2	2	2	1	1	1	0.20842	−79.28486	245.7
6	2	1	2	1	2	2	1	0.31188	−81.62437	482.4
7	2	1	3	1	1	1	2	0.20360	−79.18351	180.1
8	2	2	1	1	1	2	2	0.32368	−81.78923	255.1
9	2	2	3	2	2	2	2	0.28475	−80.88293	344.0

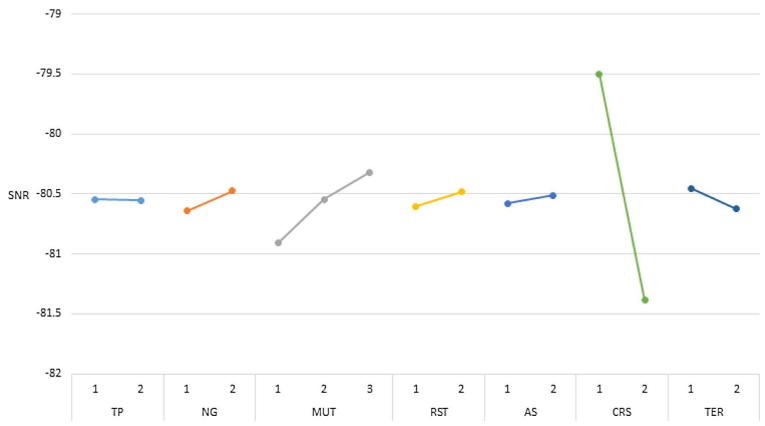

FIGURE 10. Main effects plot for S/N ratio for lowerbound deviation values.

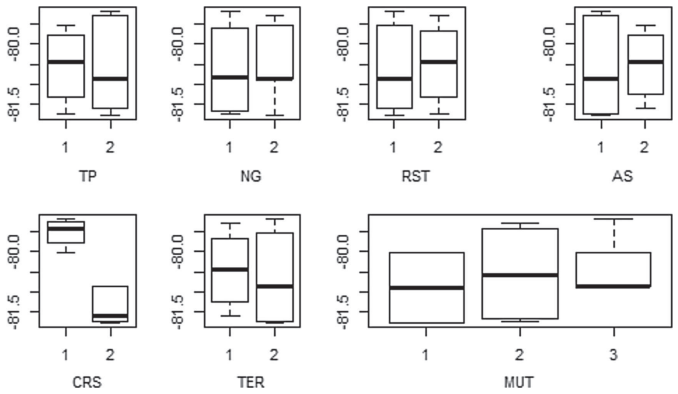

FIGURE 11. Boxplots for S/N ratio values with each factor.

TABLE 10. ANOVA table for S/N ratios for linear regression model fit considering all 7 factors.

Factor	df	Sum Sq	Mean Sq	F value	P-value	
TP	1	0.0002	0.0002	0.0125	0.9290	
NG	1	0.0640	0.0640	4.5126	0.2801	
MUT	1	0.3786	0.3786	26.6784	0.1218	
RST	1	0.0749	0.0749	5.2817	0.2613	
AS	1	0.0292	0.0292	2.0585	0.3875	
CRS	1	8.5564	8.5564	602.9946	0.0259	*
TER	1	0.0196	0.0196	1.3807	0.4489	
Residuals	1	0.0142	0.0142			
Total	8	9.1371				
Signif. codes:		"*" 0.05				

TABLE 11. ANOVA table for S/N ratio for linear regression model fit considering most significant factors.

Factor	df	Sum Sq	Mean Sq	F value	P-value	
NG	1	0.0627	0.0627	5.2218	0.08431	.
MUT	1	0.3671	0.3671	30.5578	0.00523	**
RST	1	0.0794	0.0794	6.6076	0.06195	.
CRS	1	8.5799	8.5799	714.2988	0.00001	***
Residuals	4	0.0480	0.0120			
Total	8	9.1371				
Signif. codes:		0 "***"	"**" 0.01	"*" 0.05	"." 0.1	

Adjusting the linear regression model for all seven factors and performing an ANOVA test, we observe that only CRS is statistically significant with P-value 0.0259. Then we remove, one by one, the factors whose P-value is the greatest and readjust the regression model until all factors are statistically significant obtaining the ANOVA results in Table 11.

The number of generations, mutation rate, restart, and type of crossover showed to be statistically significant, meaning that we chose the levels whose average S/N ratios are the greater. The parameter levels chosen as a result of the ANOVA test are $1000r$ generations, 0.05 of mutation rate, $\lceil 0.2r \rceil$ generations with no improvement to apply restart, and crossover type 1.

As regards to the LBD as response variable to the linear regression model. We observe in the main effects plot in Figure 12 and in boxplots in Figure 13 that LBD have a similar behavior on the control factors. However, we note that, in this case, the lower the LBD value the better.

Adjusting the linear regression model for all the seven factors and performing an ANOVA test using the LDB as response variable, we conclude that only CRS is statistically significant with P-value 0.03219. Therefore, we remove from the regression model the variables, one by one, whose P-value is the greatest and readjust the model until all factors are statistically significant achieving the ANOVA results illustrated in Table 11.

Taking into consideration the LBD as response variable to the regression model, only the mutation rate, and type of crossover are statistically significant, meaning that we would choose the mutation rate 0.05, and crossover type 1. However, these variables were already fixed at the S/N ratios analysis, and no factors that were not statistically significant for the S/N ratios showed to be statistically significant with LBD values. This leads us to choose the levels that would spend less computational time, for the factors whose level was not selected yet. Therefore, the most robust parameterization of the levels for the proposed control factors is: population size 25, $1000r$ generations, $200r$ generations with no improvement to apply restart, $100r$ pseudo-random solutions

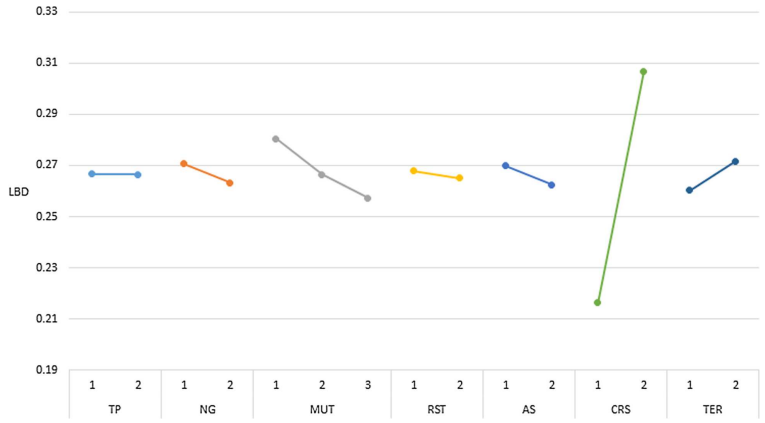

FIGURE 12. Main effects plot for lowerbound deviation.

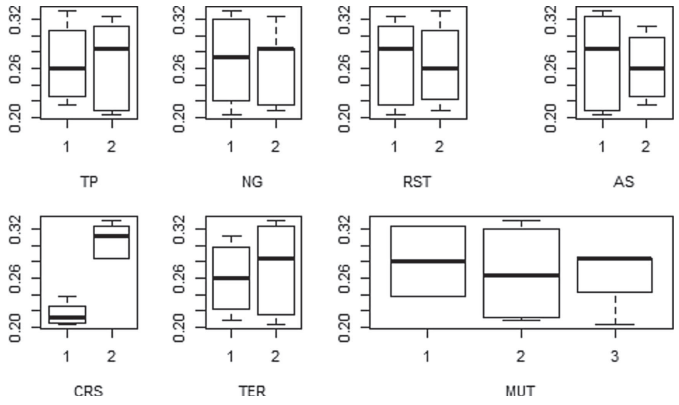

FIGURE 13. Boxplots for LBD values with each factor.

generated in the initial population and restarts, crossover type 1, 5 preserved individuals upon restart, and mutation rate of 0.05 (Tabs. 12 and 13).

6.4. Relax-and-fix heuristics

Aiming to compare our proposed GA with other methods developed to closely related cutting and packing problems, we adapt the relax-and-fix (RF) matheuristic proposed by Signorini *et al.* [17] to the problem of one-dimensional multi-period cutting stock problem for the production of precast beams. In the RF approach, a group of integer decision variables in a mixed-integer linear programming model is partitioned into mutually exclusive sets in which the number of partitions determines the number of iterations of the heuristic [25].

In each iteration, only the variables of a given partition are defined as integer decision variables, and the other decision variables are relaxed. The resulting sub-model is then solved, and two situations can occur: infeasibility and feasibility. In the case of infeasibility, the execution is stopped. In the case of feasibility, the decision variables in the previous partition are fixed in their current values, and the process is repeated for the other partitions.

We extend three RF strategies, adapted from Signorini *et al.* [17], to solve the ICP-HPBMPP. We decompose variables x_{imt} and z_t together according to dimension of periods, since they are the most numerous set of

TABLE 12. ANOVA table for LBD values for linear regression model fit considering all 7 factors.

Factors	df	Sum Sq	Mean Sq	F value	P-value	
TP	1	0.00000	0.00000	0.00340	0.96270	
NG	1	0.00012	0.00012	2.39350	0.36531	
MUT	1	0.00058	0.00058	11.48800	0.18265	
RST	1	0.00006	0.00006	1.23330	0.46669	
AS	1	0.00021	0.00021	4.22270	0.28833	
CRS	1	0.01960	0.01960	390.49990	0.03219	*
TER	1	0.00011	0.00011	2.27660	0.37261	
Residuals	1	0.00005	0.00005			
Total	8	0.02074				
Signif. codes:		"*" 0.05				

TABLE 13. ANOVA table for LBD values for linear regression model fit considering most significant factors.

Factors	df	Sum Sq	Mean Sq	F value	P-value	
MUT	1	0.00063	0.00063	6.02770	0.04944	*
CRS	1	0.01949	0.01949	187.86960	0.00001	***
Residuals	6	0.00062	0.00010			
Total	8	0.02074				
Signif. codes:		"***" 0	"*" 0.05			

variables. Variables y and o remain integer in all subproblems, since they are less numerous and cannot be partitioned according to dimension of periods.

RF1 Numbers of partitions is T, no overlap is considered, all variables in the current window have their values fixed in the next subproblem.

RF2 Numbers of partitions is T, variables are fixed only if their values are 1, then variables with value 0 are overlaped and reoptimized in the next subproblem.

RF3 Numbers of partitions if $\lceil T/3 \rceil$, no overlap is considered.

6.5. Experimental results and discussion

In order to illustrate the behavior of the genetic algorithm, we run the GA with instance cwp021. Figure 14 illustrates the best fitness and mean fitness of the populations along all generations. The *x-axis* of the Figure 14 is on logarithmic scale. The largest improvement takes place during the first generations of the GA, while in the last ones the best fitness is stagnant with some improvement upon the first restart.

In Figure 15, the best fitness values obtained by running the GA were better than CPLEX in five instances, while solutions were obtained for all instances which CPLEX could not solve.

In Figures 16 and 17, the time spent by the GA on solving each instance was significantly better than the CPLEX solution time on the large and medium-sized instances. Thus, CPLEX was faster than the GA in the small-sized instances. The *y-axis* Figure 17 is in logarithmic scale.

In Table 14, "nfsol", "nbsol", and "ave lbd" stand for number of feasible solutions, best solutions, and average lower bound deviation, respectively. Based on Table 15, we can observe that GA was the only method able to find feasible solutions for all the evaluated test instances. Aiming to provide a fair statistical comparison between the evaluated methods, we consider a lower bound deviation of 100% for a method that has not returned a feasible integer solution for a given test instance (Figs. 18 and 19).

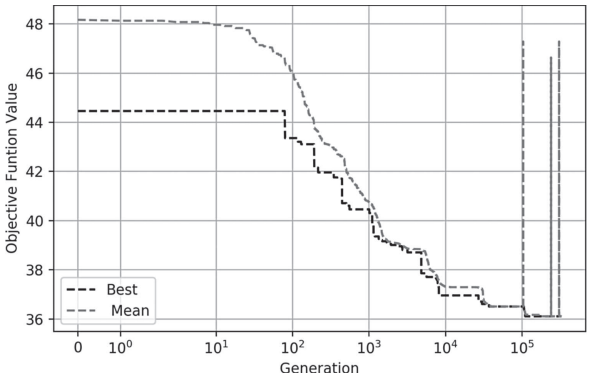

FIGURE 14. Average and best objective function value curves for instance cwp021 along generations of the selected genetic algorithm parameterization.

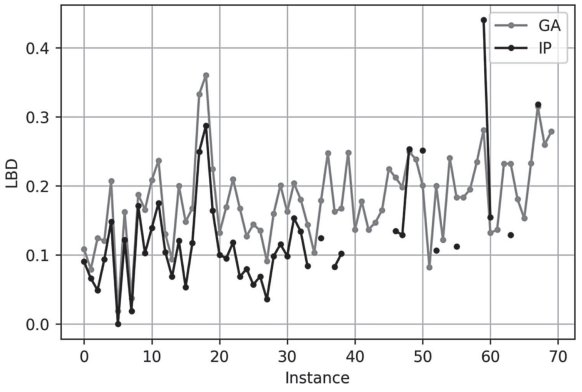

FIGURE 15. Lower bound relative deviations for CPLEX and GA with the selected parameterization.

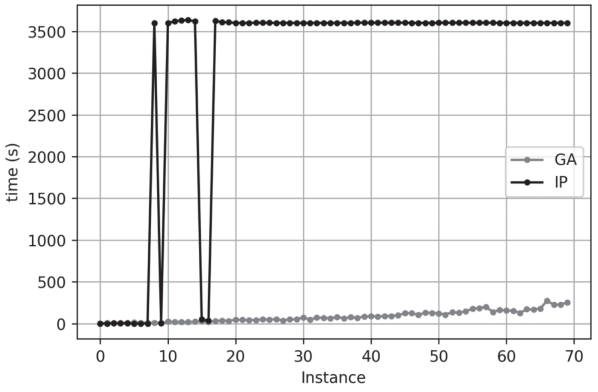

FIGURE 16. Mean time for each instance solved by CPLEX and GA with the selected parameterization.

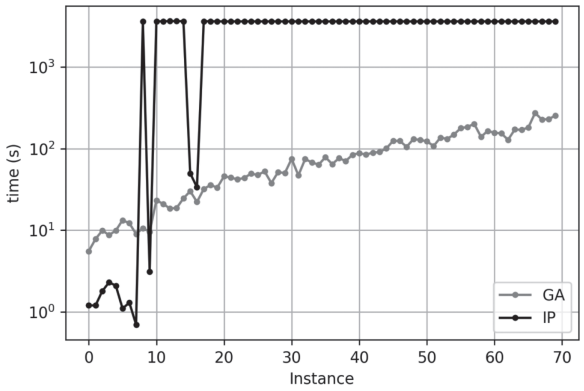

FIGURE 17. Mean time for each instance solved by CPLEX and GA with the selected parameterization, with y-axis in logarithmic scale.

TABLE 14. Overview of results with relax-and-fix methods, genetic algorithm and MILP model by instance groups.

Method		RF1	RF2	RF3	MILP	GA
All groups	nfsol	16	22	14	47	70
	nbsol	2	5	5	43	29
	ave lbd	21.38%	17.69%	14.81%	12.70%	17.36%
Group 1	nfsol	10	10	10	10	10
	nbsol	2	5	5	10	1
	ave lbd	16.70%	9.71%	14.13%	8.59%	11.97%
Group 2	nfsol	5	10	3	10	10
	nbsol	0	0	0	10	0
	ave lbd	31.21%	25.86%	15.88%	14.76%	20.35%
Group 3	nfsol	0	0	0	10	10
	nbsol	0	0	0	10	0
	ave lbd	–	–	–	8.33%	14.53%
Group 4	nfsol	1	2	1	7	10
	nbsol	0	0	0	7	3
	ave lbd	19.10%	16.81%	18.42%	11.09%	17.27%
Group 5	nfsol	0	0	0	3	10
	nbsol	0	0	0	3	8
	ave lbd	–	–	–	17.20%	18.48%
Group 6	nfsol	0	0	0	4	10
	nbsol	0	0	0	2	8
	ave lbd	–	–	–	22.74%	18.31%
Group 7	nfsol	0	0	0	3	10
	nbsol	0	0	0	1	9
	ave lbd	–	–	–	20.01%	20.62%

In order to achieve a pairwise comparison between the evaluated methods (RF1, RF2, RF3, MILP, and GA), we performed an ANOVA procedure, followed by a Tukey's test [14]. Lower bound deviations of the five methods are statistically different with p-value of 0.001578 in the ANOVA test. Figure 20 illustrates the Tukey multiple comparisons of means with 95% family-wise confidence level. Based on the results obtained, we can observe that GA outperforms all the other methods under comparison.

TABLE 15. List of notable related publications of FTP.Lower bound deviation for each test instance.

Instance	RF1	RF2	RF3	MILP	GA	Instance	RF1	RF2	RF3	MILP	GA
cwp001	9.91%	13.51%	16.22%	9.01%	9.91%	cwp036	–	–	–	12.42%	16.12%
cwp002	27.63%	6.58%	14.47%	6.58%	7.89%	cwp037	–	–	–	–	23.96%
cwp003	10.81%	6.49%	15.68%	4.86%	11.35%	cwp038	–	22.43%	–	8.23%	16.46%
cwp004	13.33%	10.67%	29.33%	9.33%	10.67%	cwp039	19.10%	11.19%	18.42%	10.15%	17.38%
cwp005	24.26%	17.16%	24.26%	14.79%	24.26%	cwp040	–	–	–	–	24.39%
cwp006	0.00%	0.00%	0.00%	0.00%	1.84%	cwp041	–	–	–	–	12.93%
cwp007	27.03%	13.51%	12.16%	12.16%	14.86%	cwp042	–	–	–	–	16.00%
cwp008	20.37%	1.85%	1.85%	1.85%	3.70%	cwp043	–	–	–	–	14.10%
cwp009	17.07%	17.07%	17.07%	17.07%	17.07%	cwp044	–	–	–	–	13.77%
cwp010	16.54%	10.24%	10.24%	10.24%	18.11%	cwp045	–	–	–	–	15.02%
cwp011	–	33.53%	–	13.87%	21.97%	cwp046	–	–	–	–	24.15%
cwp012	–	29.76%	20.79%	17.51%	21.88%	cwp047	–	–	–	13.43%	19.59%
cwp013	–	30.74%	–	10.39%	13.42%	cwp048	–	–	–	12.87%	18.93%
cwp014	40.50%	18.64%	11.11%	6.81%	10.39%	cwp049	–	–	–	25.30%	25.30%
cwp015	35.73%	25.07%	15.73%	12.00%	17.60%	cwp050	–	–	–	–	25.00%
cwp016	36.69%	13.02%	–	5.33%	8.28%	cwp051	–	–	–	25.12%	19.04%
cwp017	22.06%	22.06%	–	11.74%	16.73%	cwp052	–	–	–	–	8.36%
cwp018	–	31.23%	–	24.92%	31.89%	cwp053	–	–	–	10.62%	16.70%
cwp019	–	33.46%	–	28.68%	38.97%	cwp054	–	–	–	–	10.11%
cwp020	21.07%	21.07%	–	16.39%	22.41%	cwp055	–	–	–	–	22.96%
cwp021	–	–	–	10.00%	12.50%	cwp056	–	–	–	11.22%	17.61%
cwp022	–	–	–	9.48%	17.45%	cwp057	–	–	–	–	18.42%
cwp023	–	–	–	11.75%	19.37%	cwp058	–	–	–	–	19.46%
cwp024	–	–	–	6.81%	16.73%	cwp059	–	–	–	–	24.22%
cwp025	–	–	–	7.94%	9.96%	cwp060	–	–	–	44.00%	26.21%
cwp026	–	–	–	5.66%	12.97%	cwp061	–	–	–	15.40%	13.74%
cwp027	–	–	–	6.86%	12.39%	cwp062	–	–	–	–	13.81%
cwp028	–	–	–	3.54%	7.87%	cwp063	–	–	–	–	21.79%
cwp029	–	–	–	9.79%	16.60%	cwp064	–	–	–	12.88%	21.36%
cwp030	–	–	–	11.52%	19.52%	cwp065	–	–	–	–	16.40%
cwp031	–	–	–	9.81%	14.49%	cwp066	–	–	–	–	13.52%
cwp032	–	–	–	15.28%	20.44%	cwp067	–	–	–	–	24.12%
cwp033	–	–	–	13.35%	16.63%	cwp068	–	–	–	31.76%	29.61%
cwp034	–	–	–	8.35%	12.78%	cwp069	–	–	–	–	26.54%
cwp035	–	–	–	–	10.03%	cwp070	–	–	–	–	25.28%

Furthermore, we different methods for each instance group using Borda counting [5], as illustrated on Table 16. We note that we use Borda's original counting for handling ties. Taking this indicator into account, we can observe that the proposed GA obtained the largest count of votes of all methods under comparison, pointing to the superiority of this method.

7. FINAL REMARKS

In this work, we proposed a novel variant of cutting sequencing problems called the *integrated cutting and packing heterogeneous precast beams multiperiod production planning* (ICP-HPBMPP), which, to the best of the authors' knowledge, has not yet been studied and may have a large impact on both real-world and theoretical studies. The ICP-HPBMPP consists in integrating the problem of production planning of precast beams with the problem of cutting the traction elements used in such production, while taking into consideration the generation of leftovers and bars generated *via* overlapping.

We argued that the problem is NP-hard and proposed an integer linear programming model for its solution, in addition to a lower bound on its optimal objective function value. We also showed that restricting the

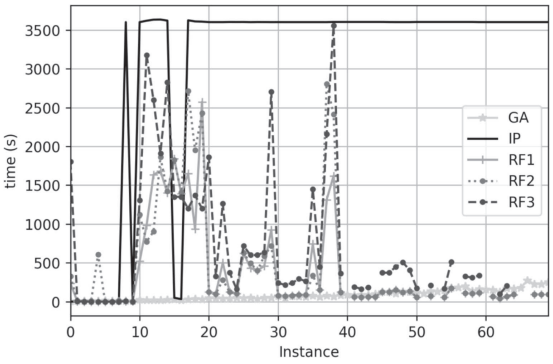

FIGURE 18. Mean time for each instance solved by CPLEX, GA, and RF methods.

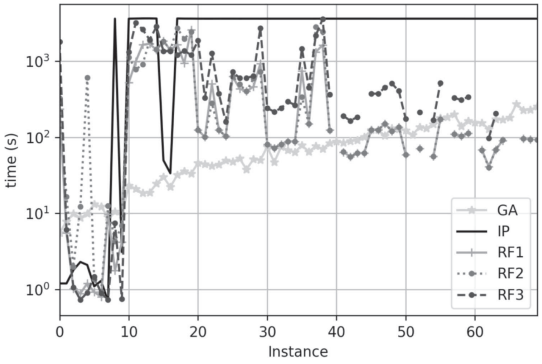

FIGURE 19. Mean time for each instance solved by CPLEX, GA, and RF methods, with y-axis in logarithmic scale.

formulation to using exclusively maximal packing patterns does not change the optimal solution set of the problem.

We also proposed three constraint programming models for generating distinct types of beam production patterns. Additionally, we introduced a set of benchmark instances and carried out computational experiments in order to evaluate the relative performance of the different solution methods studied.

The experiments showed that the integer programming model can be used to solve small size instances, while it typically does not reach optimality while solving medium-sized ones. In addition, the model usually does not find feasible solutions for large size instances. We introduced a genetic algorithm for solving the problem and fine tuned its parameters by means of a *D-optimal* experimental design to achieve improved robustness of the algorithm. The final genetic algorithm is an attractive alternative to the integer programming model, resulting in high-quality solutions in shorter solution times as compared with the exact model.

There are numerous opportunities for future work regarding the ICP-HPBMPP. In the domain of modeling, the problem can be modified to take into consideration distinct types of bars varying in matter of diameter or material, instead of only in matter of length. Also, dynamic demand could be considered, *i.e.*, in each period a new demand of beams could be included, while not exceeding a prescribed stock of bars. Regarding solution approaches, multi-objective optimization algorithms can naturally be applied to the problem, since it involves preferences between makespan and bar waste. Decomposition approaches, such as column generation, or MIP heuristics, *e.g.*, size-reduction heuristics, can also be interesting methods to be explored in conjunction with the proposed integer programming model.

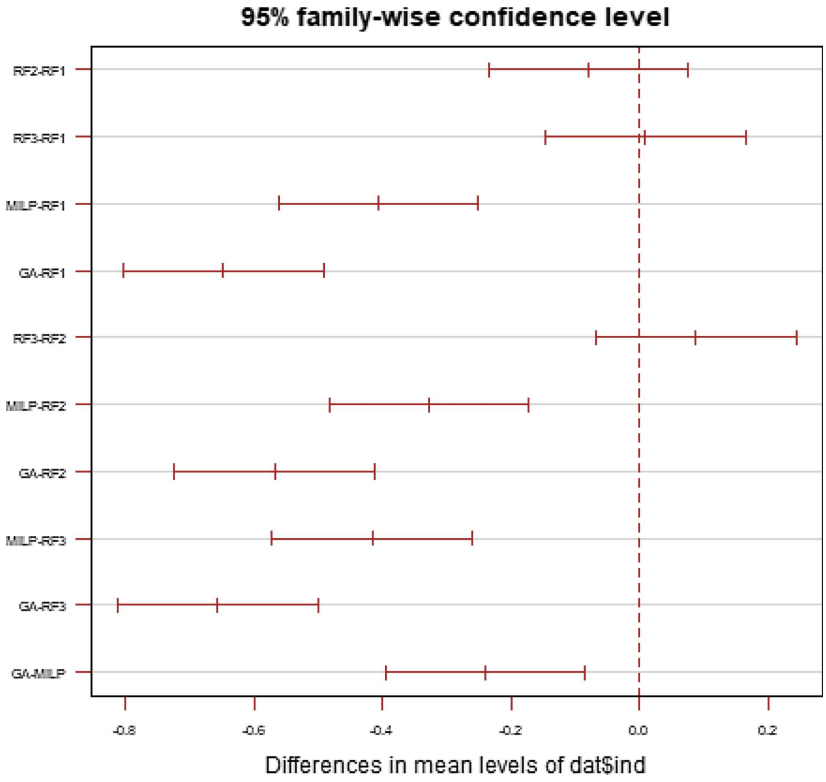

FIGURE 20. Tukey test for lower bound deviation.

TABLE 16. Borda counter points of methods for each instance group.

Group	RF1	RF2	RF3	MILP	GA
1	17	32	23	40	17
2	6	20	8	40	24
3	0	0	0	40	30
4	0	5	1	28	32
5	0	0	0	12	38
6	0	0	0	14	38
7	0	0	0	10	39
Total	23	57	32	184	218

Acknowledgements. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001. This study was financed in part by the National Council for Scientific and Technological Development (CNPq), through grant 303594/2018-7.
Conflict of interest. The authors declare that they have no conflict of interest.

REFERENCES

[1] K.A.G. Araújo, T.O. Bonates, B.A. Prata and A.R. Pitombeira-Neto, Heterogeneous prestressed precast beams multiperiod production planning problem: modeling and solution methods. *TOP* **29** (2021) 660–693.

[2] M.N. Arenales, A.C. Cherri, D.N. do Nascimento and A. Vianna, A new mathematical model for the cutting stock/leftover problem. *Pesquisa Operacional* **35** (2015) 509–522.

[3] B. Athayde Prata, A.R. Pitombeira Neto and C.J. Moraes Sales, An integer linear programming model for the multiperiod production planning of precast concrete beams. *J. Constr. Eng. Manage.* **141** (2015) 1–4.
[4] N. Beldiceanu and M. Carlsson, Global constraint catalog (Nov 2018).
[5] C. Börgers, Mathematics of Social Choice: Voting, Compensation, and Division. Society for Industrial and Applied Mathematics (SIAM) (2010).
[6] P.F. de Aguiar, B. Bourguignon, M.S. Khots, D.L. Massart and R. Phan-Than-Luu, D-optimal designs. *Chemom. Intell. Lab. Syst.* **30** (1995) 199–210.
[7] V.C. de Castilho, M.K. El Debs and M. do Carmo Nicoletti, Using a modified genetic algorithm to minimize the production costs for slabs of precast prestressed concrete joists. *Eng. App. Artif. Intell.* **20** (2007) 519–530.
[8] H. Dyckhoff, A typology of cutting and packing problems. *Eur. J. Oper. Res.* **44** (1990) 145–159.
[9] M. Gholami, M. Zandieh and A. Alem-Tabriz, Scheduling hybrid flow shop with sequence-dependent setup times and machines with random breakdowns. *Int. J. Adv. Manuf. Technol.* **42** (2009) 189–201.
[10] P.C. Gilmore and R.E. Gomory, A linear programming approach to the cutting-stock problem. *Oper. Res.* **9** (1961) 849–859.
[11] P.C. Gilmore and R.E. Gomory, A linear programming approach to the cutting stock problem – part II. *Oper. Res.* **11** (1963) 863–888.
[12] R.T. Marler and J.S. Arora, Survey of multi-objective optimization methods for engineering. *Struct. Multi. Optim.* **26** (2004) 369–395.
[13] G.M. Melega, S.A. de Araújo and R. Jans, Classification and literature review of integrated lot-sizing and cutting stock problems. *Eur. J. Oper. Res.* **271** (2018) 1–19.
[14] D.C. Montgomery, Design and Analysis of Experiments. John Wiley & Sons (2017).
[15] J.J. Pignatiello Jr, An overview of the strategy and tactics of taguchi. *IIE Trans.* **20** (1988) 247–254.
[16] K.C. Poldi and M.N. Arenales, O problema de corte de estoque unidimensional multiperíodo. *Pesquisa Operacional* **30** (2010) 153–174.
[17] C.D.A. Signorini, S.A. de Araújo and G.M. Melega, One-dimensional multi-period cutting stock problems in the concrete industry. *Int. J. Prod. Res.* (2021) 1–18. DOI: 10.1080/00207543.2021.1890261.
[18] H. Stadtler, A one-dimensional cutting stock problem in the aluminium industry and its solution. *Eur. J. Oper. Res.* **44** (1990) 209–223.
[19] F. Triefenbach, *Design of experiments: the D-optimal approach and its implementation as a computer algorithm*. Bachelor's Thesis in Information and Communication Technology (2008).
[20] P. Trkman and M. Gradisar, One-dimensional cutting stock optimization in consecutive time periods. *Eur. J. Oper. Res.* **179** (2007) 291–301.
[21] P.H. Vance, Branch-and-price algorithms for the one-dimensional cutting stock problem. *Comput. Optim. App.* **9** (1998) 211–228.
[22] A.H.D. Vassoler, S.C. Poltroniere and S.A. Araújo, Modelagem matemática para o problema de produção de vigotas na indústria de lajes treliçadas. *Revista Eletrônica Paulista de Matemática* **7** (2016) 68–77.
[23] Z. Wang, H. Hu and J. Gong, Framework for modeling operational uncertainty to optimize offsite production scheduling of precast components. *Autom. Constr.* **86** (2018) 69–80.
[24] G. Wäscher, H. Haußner and H. Schumann, An improved typology of cutting and packing problems. *Eur. J. Oper. Res.* **183** (2007) 1109–1130.
[25] L.A. Wolsey, Integer Programming. John Wiley & Sons (2020).