UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE ESTATÍSTICA E MATEMÁTICA APLICADA

PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM E MÉTODOS QUANTITATIVOS

MESTRADO ACADÊMICO EM MODELAGEM E MÉTODOS QUANTITATIVOS

KENNEDY ANDERSON GUIMARÃES DE ARAÚJO

CONTRIBUTIONS TO THE MULTIPERIOD PRODUCTION PLANNING OF HETEROGENEOUS PRECAST BEAMS

FORTALEZA

2019

KENNEDY ANDERSON GUIMARÃES DE ARAÚJO

CONTRIBUTIONS TO THE MULTIPERIOD PRODUCTION PLANNING OF HETEROGENEOUS PRECAST BEAMS

Dissertação apresentada ao Curso de Mestrado Acadêmico em Modelagem e Métodos Quantitativos do Programa de Pós-Graduação em Modelagem e Métodos Quantitativos do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Modelagem e Métodos Quantitativos. Área de Concentração: Inteligência Computacional e Otimização

Orientador: Prof. Dr. Tibérius de Oliveira e Bonates

Co-Orientador: Prof. Dr. Bruno de Athayde Prata

FORTALEZA

2019

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

A689c Araújo, Kennedy Anderson Guimarães de.
Contributions to the Multiperiod Production Planning of Heterogeneous Precast Beams / Kennedy Anderson Guimarães de Araújo. – 2019.
109 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Modelagem e Métodos Quantitativos, Fortaleza, 2019.
Orientação: Prof. Dr. Tibérius de Oliveira e Bonates.
Coorientação: Prof. Dr. Bruno de Athayde Prata.

1. Cutting and packing problems. 2. Integer linear programming. 3. Combinatorial optimization. I. Título.

CDD 510

KENNEDY ANDERSON GUIMARÃES DE ARAÚJO

CONTRIBUTIONS TO THE MULTIPERIOD PRODUCTION PLANNING OF HETEROGENEOUS PRECAST BEAMS

Dissertação apresentada ao Curso de Mestrado Acadêmico em Modelagem e Métodos Quantitativos do Programa de Pós-Graduação em Modelagem e Métodos Quantitativos do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Modelagem e Métodos Quantitativos. Área de Concentração: Inteligência Computacional e Otimização

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Tibérius de Oliveira e Bonates (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Bruno de Athayde Prata (Co-Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Jesus Ossian da Cunha
Universidade Federal do Ceará (UFC)

Prof. Dr. Martín Gómez Ravetti
Universidade Federal de Minas Gerais (UFMG)

Dedico esta dissertação aos meus pais. Obrigado por tudo!

AGRADECIMENTOS

Agradeço, primeiramente, aos meus pais que com toda a dificuldade que tive nesses dois anos de mestrado estiveram sempre ao meu lado e me apoiaram em todos os instantes desta jornada de todas as formas possíveis. Os quais responsabilizo por todo o sucesso que obtive e cada conquista que realizei no período do curso e em toda minha vida.

Aos meus orientadores Tibérius de Oliveira e Bonates e Bruno de Athayde Prata por toda a assistência, incansável disponibilidade para apoiar-me e por todo o tempo que dedicaram a esclarecer dúvidas e a me ajudar durante o processo de realização deste trabalho.

Aos membros da "Matilha Aleatória" por serem as pessoas incríveis que são, por me fazerem tão bem à minha saúde mental, e pela amizade e companheirismo. Em especial, agradeço ao Deni por sua incansável disponibilidade e toda a ajuda que me deu neste trabalho e nas disciplinas do curso.

Aos meus amigos de mestrado, Rossana, Lívia, Armando, Vinícius, e Raul, pela imensa ajuda e amizade ao longo de todo o curso. Aos amigos Lucas, Samuel, e Nilton, pelo companheirismo e por tornarem os últimos meses de curso muito mais agradáveis e divertidos.

Ao corpo docente e administrativo do curso pela colaboração, esclarecimentos, e disponibilidade de ajudar, e a todos que contribuíram indiretamente para a realização deste trabalho.

Shadows will scream that I'm alone
But I know, we've made it this far, kid
(Migraine, Twenty One Pilots)

ABSTRACT

In this work, we introduce two novel variants of cutting scheduling problems named Heterogeneous Prestressed Precast Beam Multiperiod Production Planning (HPPBMPP) and Integrated Cutting and Packing Heterogeneous Precast Beam Multiperiod Production Planning (ICP-HPBMPP). Concrete precast beams are those which are cast away from the construction site in a controlled environment and ideal conditions, whilst a prestressed precast beam is a type of concrete precast beams that is stretched with traction elements in order to improve its resistance and behavior in service. Both kinds of beams can be of different lengths, types, and potentially require different curing times. The HPPBMPP consists of planning the usage of the available set of molds within a given time horizon to fulfill a given demand of prestressed precast beams. On the other hand, the ICP-HPBMPP addresses the HPPBMPP applied to precast beams integrated to the cutting phase of bars that are used in the production of such beams. In this scenario, one must take into consideration the generation and the use of leftovers, as well as the possibility of dealing with overlapping bars, i.e., bars that are assembled by connecting two existing bars of smaller sizes. We propose integer linear programming (ILP) models for both problems, in addition to alternative solution methods, such as size-reduction heuristics, priority rules, and genetic algorithms. We argue the NP-hardness of both problems and explore some of their properties, including lower bounds for optimal objective function values and the use of maximal patterns. We discuss the results of computational tests with the exact solution of the ILP models and the alternative solution methods proposed. We conclude with a discussion of the relative merits of the proposed approaches in terms of solution quality. We infer that the proposed size reduction heuristic and genetic algorithms are good alternatives to ILP models producing good solutions with lower computing time for both problems.

Keywords: Cutting and packing problems; Integer linear programming; Combinatorial optimization.

RESUMO

Neste trabalho, apresentamos duas novas variantes de problemas de sequenciamento de corte chamados de Planejamento de Produção Multiperíodo de Vigas Pré-Moldadas Protendidas Heterogêneas (HPPBMPP em inglês) e Problema Integrado de Corte e Empacotamento do Planejamento de Produção Multiperíodo de Vigas Pré-Moldadas Heterogêneas (ICP-HPBMPP em inglês). Vigas pré-moldadas de concreto são aquelas que são moldadas longe do local de construção em um ambiente controlado e sob condições ideais, enquanto uma viga pré-moldada protendida é um tipo de viga pré-moldada de concreto que é tensionada com elementos de tração para melhorar sua resistência e comportamento em serviço. Ambas as classes de viga podem ter comprimentos e tipos diferentes, e, potencialmente, requerer diferentes tempos de cura. O HPPBMPP consiste em planejar o uso do conjunto disponível de formas dentro de um determinado horizonte de tempo para atender a uma dada demanda de vigas pré-moldadas protendidas. Por outro lado, o ICP-HPBMPP aborda o HPPBMPP aplicado a vigas pré-moldadas integrado à fase de corte de barras, que são utilizadas na produção de tais vigas. Neste cenário, pode-se levar em consideração a geração e o uso de sobras, assim como a possibilidade de lidar com barras produzidas por traspasse, isto é, barras que são montadas por meio de uma conexão de duas barras existentes de tamanho menor. Propomos modelos de programação linear inteira (ILP em inglês) para ambos os problemas, além de métodos alternativos, como heurísticas de redução de tamanho, regras de prioridade e algoritmos genéticos. Argumentamos a NP-dificuldade de ambos os problemas e exploramos algumas de suas propriedades, incluindo limites inferiores para valores ótimos de função objetivo e o uso de padrões maximais. Discutimos os resultados de testes computacionais com a solução exata dos modelos de ILP e os métodos alternativos propostos. Concluímos com uma discussão dos méritos relativos das abordagens propostas em termos de qualidade da solução. Inferimos que a heurística de redução de tamanho e algorítimos genéticos propostos são boas alternativas aos modelos de ILP produzindo boas soluções com menor custo computacional para ambos os problemas.

Palavras-chave: Problemas de corte e empacotamento; Programação linear inteira; Otimização Combinatória.

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

CONTENTS

1 INTRODUCTION

Precast beams are common in factories of civil construction materials, since they are often used in a variety of construction types. They are preferred over steel beams, since concrete has a low price and requires less maintenance when compared to steel. Also, the execution of such structure type is simple, what makes its use frequent. Unlike conventional precast beams production, a traction process is used in the production of prestressed precast beams. This process aims at improving the resistance and behavior of the beams in service. For their production, a factory uses concrete, along with the traction elements and a set of reusable molds. Aside from length, such kind of beams can also vary in curing time and type. A type of beam cannot be produced in the same mold on the same period of a beam of another type. The Heterogeneous Prestressed Precast Beams Multiperiod Production Planning Problem (HPPBMPP) consists in finding a feasible production planning to cast certain quantities of prestressed precast concrete beams, possibly of different types, while optimizing certain objective function.

The HPPBMPP is a combinatorial problem that arises in practical scenarios. Finding an optimal solution to the problem can become a challenging task, as soon as parameters such as the numbers of beam types, lengths and demands increase beyond trivial values. Nevertheless, and despite the similarities between the HPPBMPP and cutting problems (see Wäscher *et al.* (2007) for a typology), the problem does not precisely fit any existing formulation in the field of combinatorial optimization, to the best of our knowledge.

There are several studies about Multiperiod One-dimensional Cutting Stock Problems (M1DCSP), which is a particular case of the HPPBMPP, in the literature. Trkman and Gradisar (2007) proposed a model for the M1DCSP considering the use of objects/leftovers in stock. Poldi and Arenales (2010) proposed an integer linear model for the M1DCSP, implemented a simplex method with column generation to solve the linear relaxation, and developed two rounding heuristics for finding integer solutions the problem. Prata *et al.* (2015) introduced a special case of the HPPBMPP and proposed an integer programming model for the multiperiod production planning of precast concrete beams. The proposed model, however, handled the simplest case, in which all beams are homogeneous. Arenales *et al.* (2015) proposed a mathematical model to the cutting stock/leftover problem and suggested a column generation technique to solve the linear relaxation of the problem. Vassoler *et al.* (2016) proposed a mathematical model based on multiperiod cutting stock problem for the production planning problem of joists in trusses slabs industries. The authors proposed a solution method based on column generation, based on

(GILMORE; GOMORY, 1961) and (GILMORE; GOMORY, 1963), to solve the linear relaxation of the problem. Aiming at introducing new variants of the M1DCSP that do not completely fit the problems described in the literature, we present this dissertation divided into two parts:

1. We introduce the HPPBMPP, which is a novel variant of the M1DCSP. We propose two models based on integer linear programming in order to optimize the HPPBMPP by seeking a solution that either minimizes the losses or minimizes the number of periods to produce the demand. This material is presented in the form of an individual article.
2. We introduce a novel variant of the HPPBMPP, called the Integrated Cutting and Packing Heterogeneous Precast Beam Multiperiod Production Planning (ICP-HPBMPP), considering the integration of the cutting process of bars, which are used for the production of precast beams. This material is also presented in the form of an individual article.

The two novel problems and innovative approaches for their solution presented in the two parts of this dissertation are tools to support decision making in precast beam production. The proposed models allow the identification of minimum loss and minimum time solutions in an automated process, resulting in reductions of planning times as well as allowing scenario studies.

2 HETEROGENEOUS PRESTRESSED PRECAST BEAMS MULTIPERIOD PRODUCTION PLANNING PROBLEM: MODELING AND SOLUTION METHODS

Abstract

A prestressed precast beam is a type of beam that is stretched with traction elements. A common task in a factory of prestressed precast beams involves fulfilling, within a time horizon, the demand ordered by clients. A typical order includes beams of different lengths and types, with distinct beams potentially requiring different curing periods. We refer to the problem of planning such production as Heterogeneous Prestressed Precast Beams Multiperiod Production Planning (HPPBMPP). We formally define the HPPBMPP, argue its NP-hardness, and introduce four novel integer programming models for its solution and a size reduction heuristic (SRH). We propose six priority rules to produce feasible solutions. We perform computational tests on a set of synthetic instances that are based on data from a real-world scenario and discuss a case study. Our experiments suggest that the models can optimally solve small instances, while the SRH can produce high-quality solutions for most instances.

Keywords: prestressed concrete; precast beams; modular construction; integer linear programming; size reduction heuristics.

2.1 Introduction

Unlike conventional precast beams, prestressed precast beams have a different production process, in which they are tensioned using traction elements prior to supporting any actual load. The aim of this process is to improve the resistance and behavior of the beams in service. For their production, a factory uses concrete, along with the traction elements and a set of reusable molds. Traction elements are positioned and tensioned within the molds, after which concrete is cast. This is followed by a curing period, during which the concrete bonds to the traction elements. Those elements are then released and, as their material attempts to resume its original (untensioned) length, the concrete is compressed due to static friction. Prestressed beams are common in factories of civil construction materials, since they are often used in a variety of construction types. They are preferred over steel beams, since concrete has a low price and requires less maintenance when compared to steel. For the purpose of this chapter, prestressed precast beams can vary with respect to curing time, length, and the number of traction elements used.

A common task in this type of factory involves fulfilling the demand of a set of clients, within a given time horizon. A typical order includes beams of different lengths and types, with different types of beams potentially requiring different curing times. A mold can be used to produce several beams simultaneously, with the total length of the beams being limited by the mold's capacity. While a given mold can be used to produce different types of beams in different periods, only one type of beam can be produced at a given mold at any given time. The problem of planning such production while minimizing the idle capacity in the molds will be referred to as the Heterogeneous Prestressed Precast Beams Multiperiod Production Planning (HPPBMPP).

To the best of the authors' knowledge, the HPPBMPP is novel, despite its similarity with existing cutting problems. Indeed, we argue that the problem includes a known NP-hard cutting problem as a particular case. The combinatorial nature of the problem makes it hard for managers to generate good schedules in practice, which results in inefficiencies and delays in production. The practical importance of the problem also derives from the high-performance, durability, and versatility of prestressed precast beams. Those factors are responsible for the frequent use of such beams in a number of building types and civil structures, ranging from houses and office buildings to bridges and dams. Optimizing the production of prestressed beams has the potential effect of speeding up overall construction time, while improving the usage of molds, allowing factories to accept additional orders due to shorter lead times.

In this chapter, four integer linear programming models for the multiperiod production pĺanning of prestressed precast concrete beams of multiple types are presented. The first model maximizes the usage of molds, while the second model minimizes the makespan of the entire production, the third model consists in a alternative model to minimize the makespan, and the fourth model and last one minimizes the total completion time. A set of benchmark instances for the problem are also introduced, being derived from data of a real-life application. The remainder of this chapter is organized as follows. In Section 2.2, the related literature is reviewed. In Section 2.3, the HPPMBPP problem is formally defined, we argue that it is NP-hard, and the four mathematical programming models and specific sets of patterns are described. In Section 2.4 a total of 6 priority rules are proposed. In Section 2.5, the instances used in this chapter are characterized, an auxiliary constraint programming model for the generation of production patterns is outlined and the computational performance of the models using different sets of patterns is evaluated. In Section 2.6 a case study is solved and its solutions are analyzed. In

Section 2.7, some conclusions are derived and opportunities for further research are discussed.

2.2 Related work

To the best of our knowledge, the HPPBMPP problem has not been previously studied in the revised literature. A special case of the problem has been introduced in (PRATA *et al.*, 2015) and an integer programming model has been proposed for its solution.The authors argued that the problem is closely related to cutting stock and sequencing problems, both of which have been extensively investigated. We bring attention to the following similarities between those problems and the HPPBMPP:

1. In the HPPBMPP setting, a mold can represent a large beam of a certain type that must be cut into smaller pieces, with each piece corresponding to the beams that are produced in the mold. In this interpretation, the leftover part of the large beam corresponds to the mold's unused capacity, rather than actual wasted material;
2. In HPPBMPP, the production might require several periods before the entire demand has been met, i.e., before all beams have been produced. Producing different beam types may require different curing times. The usage of the molds must be scheduled in such a way as to avoid overlapping (the same mold being used to simultaneously produce different types of beams), while respecting the maximum time allowed, or while minimizing some notion of tardiness.

Cutting stock and sequencing are among the most studied problems in the operations research literature. The one-dimensional cutting problem, in particular, bears close resemblance to the HPPBMPP, in the sense that the production in each mold can be planned (equivalently, the mold can be "cut") independently of other molds. In what follows, we highlight some studies that tackle scheduling and cutting problems, and that we consider relevant to our study.

A variety of heuristic methods have been successfully applied to those two classes of problems. Yuen (1991) suggested two heuristics for sequencing cutting patterns in the Australian glass industry and reported substantial savings and low computing times. Wäscher and Gau (1996) studied the computational performance of heuristics for the one-dimensional cutting stock problem that work by exploring the neighborhood of an optimal solution to the linear relaxation of a model. The heuristics were reported to find optimal solutions for the majority of the instances tested. Shahin and Salem (2004) presented a genetic algorithm (GA) for solving the one-dimensional cutting stock problem. The authors also studied three real-life scenarios

arising from a steel workshop and compared the solutions (cutting schedules) obtained by their algorithm with the actual workshop cutting schedules. Pileggi *et al.* (2005) presented three heuristic approaches to deal with an integrated pattern generating and sequencing problem. The authors considered the trade-off between the different objective functions involved and compared them in the one-dimensional cutting case. Benjaoran *et al.* (2005) proposed a multi-objective flow shop scheduling model for bespoke precast concrete production planning and used a genetic algorithm for its solution. Benjaoran and Bhokha (2014) developed new solution procedures for finding efficient cutting plans while minimizing trim loss and the number of stocks used for the cutting stock problem of construction steel bars.

Studies that are solely based on exact methods as a solution procedure have also been reported. Arenales *et al.* (2015) proposed a new mathematical model for the cutting stock/leftover problem (CSLP). Due to the exceedingly large size of the model, the authors proposed to solve its linear relaxation via column generation and to use heuristics for constructing feasible solutions based on the relaxed solution. Braga *et al.* (2016) explored an exact and compact assignment formulation for the combined cutting stock and scheduling, along with valid inequalities that are used with a cutting-plane algorithm.

Another fruitful line of work involves the use of both heuristic and exact methods in a combined solution approach. For instance, Yanasse and Lamosa (2007) solved to optimality an integrated problem that involved a cutting stock problem under particular pattern sequencing constraints. Their approach included an integer linear programming (ILP) model, a proposed decomposition scheme to solve the model, a modified subgradient method to solve the dual problem, and several heuristic algorithms. Gramani and França (2006) formulated a mixed-integer mathematical model for solving the combined cutting stock and lot-sizing problem in a multi-period planning scenario. The authors proposed a heuristic method based on a shortest path algorithm to minimize trim loss. Nonas and Thorstenson (2008) proposed a new column generating solution procedure for the combined cutting-stock and lot-sizing problem, combined with tree-like and sequential heuristics. Salem *et al.* (2007) presented three approaches for solving the one-dimensional cutting stock problem: a genetic algorithm, a linear programming model, and an ILP model. The authors studied three real-life case studies from a steel workshop. Arbib and Marinelli (2014) proposed an exact ILP formulation for the cutting stock problem with due dates with the aim of minimizing a combination of the number of objects cut and weighted tardiness. The authors developed primal heuristics, upper bounds, and an implicit enumeration

scheme.

The production of precast items has also been previously considered from the optimization viewpoint. As an example, Shih and Liu (2010) optimized a production project of precast items via a mixed integer linear programming model based on grouping concepts and a recursive procedure. Ko and Wang (2011) approached the problem of scheduling precast production considering six steps: mold assembly, placement of reinforcement and all embedded parts, concrete casting, curing, mold stripping, and product finishing. The authors developed a mathematical model and a multi-objective genetic algorithm to solve it. Khalili and Chua (2013) dealt with the optimization of resources and costs for the precast production of complex configurations by means of a mixed ILP model based on prefabrication configuration and component grouping ideas. Yang *et al.* (2016) made a study in precast production proposing a model for the Flowshop Problem of Multiple Production Lines and developed a genetic algorithm for the problem optimization. The authors identify several objective functions and optimization constraints, although only the optimization objective of makespan minimization was used to simplify the comparisons of the proposed approach. Chen *et al.* (2017) proposed an ILP model for optimizing precast production planning, allocation of component storage, and transportation, as well as for making timely adjustments for contracted projects, with the aim of minimizing production costs.

Additionally, some studies have tackled the cutting stock problem considering due dates or multiple periods. For example, Li (1996) developed heuristics and two two-dimensional cutting stock models with due date and release date constraints, in which meeting orders' due dates are more important than minimizing the waste of materials. Nonås and Thorstenson (2000) proposed a non-linear optimization model for the combined cutting-stock and lot-sizing problem and suggested several heuristics for finding feasible solutions. Reinertsen and Vossen (2010) proposed new optimization models for solving the cutting stock problem when orders have due dates. The authors solved the models via column generation, with the corresponding pricing problems solved with shortest path algorithms. Prata *et al.* (2015) proposed an integer programming model for the multi-period production planning of precast concrete beams. The proposed model, however, handled the simplest case of the HPPBMPP, in which all beams are of the same type, or, equivalently, have unitary curing time.

2.3 Problem statement

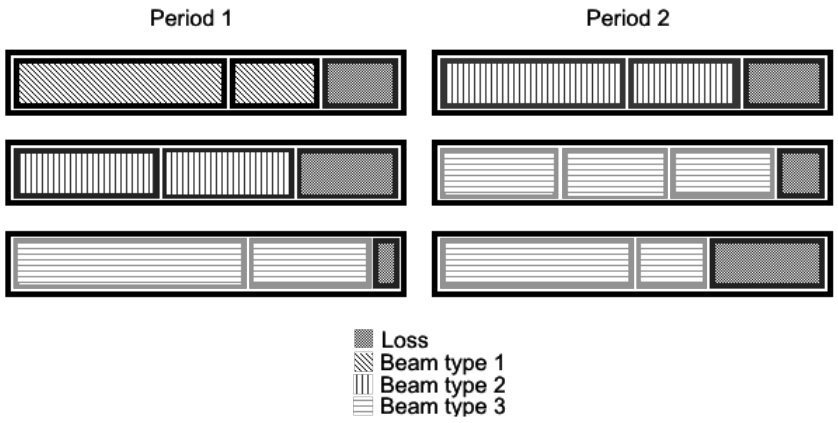
The HPPBMPP consists of planning the usage of the available molds along a given time horizon, i.e., scheduling the beam production in such molds, to cast a demand of prestressed precast concrete beams, possibly of different types, while minimizing the total unused capacity of the molds, i.e. the total idle capacity. In order to formalize the problem, we present the input of the HPPBMPP 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 $c = 1,\ldots,C$;
- $l(c,k)$: real numbers corresponding to the actual lengths of beams of type c, with $c = 1,\ldots,C$ and $k = 1,\ldots,q_c$;
- $d(c,k)$: 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 the m-th mold, with $m = 1,\ldots,M$.

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 the m-th mold cannot be greater than L_m and the total number of days required to complete the production cannot be greater than T. The idle capacity of the m-th mold at the t-th period, given by $\mathscr{I}(m,t)$, is the difference between the total length of beams produced in the m-th mold and its capacity. If the m-th mold is not used for beam production in the t-th period $\mathscr{I}(m,t)$ is zero. The HPPBMPP output consists of a production plan that minimizes the sum of idle capacities over all molds and periods, i.e., the minimum possible value of $\sum_{t=1}^{T}\sum_{m=1}^{C}\mathscr{I}(m,t)$ that can be achieved while fulfilling the demand of beams. An example of a feasible production plan is shown in Figure 1.

The HPPBMPP is a combinatorial problem that arises in practical scenarios. Finding an optimal solution to the problem can become a challenging task, as soon as parameters such as the numbers of beam types, lengths and demands increase beyond trivial values. Nevertheless, and despite the similarities between the HPPBMPP and cutting problems, the problem does not precisely fit any existing formulation in combinatorial optimization. Note that, for the purpose

Figure 1 – Example of feasible solution. Three beam types, of various lengths, produced in three molds



of this work, we do not consider neither delivery dates nor stock control in the HPPBMPP and the parameter T is an estimate of the time horizon needed to produce the total demand of beams. We can, however, establish the hardness of the problem:

Proposition 2.3.1. *The HPPBMPP is NP-hard.*

The assertion in Proposition 2.3.1 can be made due to the fact that HPPBMPP includes, as a particular case, the classical one-dimensional cutting stock problem. Indeed, the case in which there exists only one beam type (i.e., all beams have the same number of cables and the same curing time) turns out to be precisely an instance of the one-dimensional cutting stock problem: the items to be cut correspond to the molds, while the waste of material is equivalent to the unused capacity of each mold. The 1-dimensional cutting stock problem has been known to be NP-hard from the fact that the knapsack problem is reducible to it (GAREY *et al.*, 1979).

In this section, four models that extend the model proposed by Prata *et al.* (2015) are described for the case of multiple beam types. In this scenario, different beam types can demand different curing times, unlike the problem treated by Prata *et al.* (2015). Moreover, a mold cannot be used to simultaneously produce beams of different types.

If beams of type c are produced in the m-th mold at a given period, it is possible to describe the current state of the mold as a non-negative integer tuple $(a_1, a_2, \ldots, a_{q_c})$, with each a_k $(1 \leq k \leq q_c)$ specifying the quantity of beams of length $l(c,k)$ that are currently being produced in the mold. The information of the beam type and the tuple that describes the quantity of each beam length produced — i.e., the pair $\big(c, (a_1, a_2, \ldots, a_{q_c})\big)$ — will be called a *pattern*. Naturally, only patterns that do not exceed the m-th mold capacity can be produced in that mold.

Thus, as a practical matter, we can limit ourselves to taking into consideration only patterns that do not exceed the largest capacity among the molds in the problem's data.

A solution for the HPPBMPP requires fully specifying the pattern that is used in each mold during each of the T periods, with the same pattern being potentially used more than once. The existence of a special pattern P_0 is assumed, 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. Since the curing time of each beam type can be different, it is necessary to include constraints in the model that identify the patterns associated with the consecutive periods during which a particular pattern is under production. When our model selects pattern $P_i = \big(c, (a_1, a_2, \ldots, a_{q_c})\big)$, with $c = 1, \ldots, C$, to be initiated in the m-th mold at period t, it will accordingly select pattern P_0 to be used in that mold during the subsequent periods $t+1, \ldots, t+t_c-1$.

For instance, consider that a mold m is used to initiate the production of beams of curing time 3 at period 5. Then, the pattern corresponding to the production of those beams must be assigned to period 5 while P_0 must be assigned to the subsequent periods in m: 6 and 7. This fully describes the state of the mold during periods 5, 6, and 7.

In order to refer to specific information on a given pattern $P_i = \big(\bar{c}, (\bar{a}_1, \ldots, \bar{a}_{q_c})\big)$, with $\bar{c} = 1, \ldots, C$, 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 $c = \bar{c}$, then $\mathcal{N}_i(c,k) = \bar{a}_k$, with $k \in \{1, \ldots, q_c\}$; otherwise, $\mathcal{N}_i(c,k) = 0$, for any k.
- $u(P_i)$: capacity used by P_i, i.e. $u(P_i) = \sum_{k=1}^{q_{\bar{c}}} l(\bar{c},k) \cdot P_i(\bar{c},k)$.
- E_i: number of periods required to produce the beams in P_i. 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 $\bar{c}$, given by $t_{\bar{c}}$.
- F_i^m: idle capacity of the m-th mold when pattern P_i is used in that mold. Note that this quantity depends on the lengths of the beams specified in the pattern, the mold capacity, and the value of E_i. F_i^m can be computed as follows: $E_i \cdot (L_m - u(P_i))$. For instance, if the capacity of the m-th mold is 10, the capacity used by pattern P_i is 6, and $E_i = 3$, then we have $F_i^m = 3 \cdot (10-6) = 12$.

Both E_i e F_i^m can be directly calculated from the problem's data. The value of F_0^m, associated to the P_0, is defined as zero. However, its is possible to envision variants of the formulation proposed here, in which alternative values for F_0^m are used, depending on the

particular objective function to be optimized.

A remark concerning the use of the P_0 pattern is in order. 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. Moreover, this type of situation is not regarded as a loss. On the other hand, when a mold is used to initiate the production of pattern P_i at period t, the subsequent $E_i - 1$ periods are assigned P_0. This situation results in a total loss of F_i^m, which corresponds to the unused capacity of the mold, multiplied by the number of days required for the production of P_i.

Given a set of patterns $\{P_1, \ldots, P_r\}$, not including P_0, we define the following sets:

- $Q(m)$: set containing the indices of the patterns 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 $m = 1, \ldots, M$. The same pattern can be used in different molds of potentially distinct lengths.
- $S(j)$: set of indices of the patterns that have curing time $j \in \{1, ..., R\}$, with R being the largest curing time of all beam types present in the problem instance.
- $Q^*(m)$: set $Q(m)$ including pattern P_0, i.e. $Q^*(m) = Q(m) \cup \{0\}$.

Our models involve the binary decision variables $x_i^{m,t}$, for $i = 1, \ldots, r$, $m = 1, \ldots, M$, $t = 1, \ldots, T$, each of which is associated with the use of pattern P_i in the m-th mold during period t, as follows:

$$x_i^{m,t} = \begin{cases} 1, & \text{if pattern } P_i \text{ is initiated in the } m\text{-th mold at period } t; \\ 0, & \text{otherwise.} \end{cases}$$

In a scenario of uninterrupted production, exceeding the prescribed demand is usual, although to keep a stock of spare beams can be expensive and limited physically. In addition, in a real-life scenario, it might be desirable to use only patterns that have a minimal percentage of occupation of the molds. If we limit ourselves to using those types of patterns, it may become impossible to satisfy the demands at equality (it could be necessary to use extremely simple patterns to achieve equality). In view of that, the model presented next satisfies the demand with the possibility of surplus. Therefore, the choice of satisfying demands with the possibility of excess seems to be of practical value.

2.3.1 Model for minimizing idle capacity

We now introduce our main model for the HPPBMPP as follows:

(M1) minimize

$$\sum_{m=1}^{M} \sum_{i \in Q(m)} \sum_{t=1}^{T} F_i^m \, x_i^{m,t} \tag{2.1}$$

subject to

$$\sum_{i \in Q^*(m)} x_i^{m,t} \leq 1, \qquad m = 1, \ldots, M, \quad t = 1, \ldots, T \tag{2.2}$$

$$\sum_{m=1}^{M} \sum_{i \in Q(m)} \sum_{t=1}^{T-E_i+1} \mathscr{N}_i(c,k) \, x_i^{m,t} \geq d(c,k), \qquad k = 1, \ldots, q_c, \quad c = 1, \ldots, C \tag{2.3}$$

$$(E_i - 1) \, x_i^{m,t} \leq \sum_{\alpha=1}^{E_i-1} x_0^{m,t+\alpha}, \qquad m = 1, \ldots, M, \quad t = 1, \ldots, T - E_i + 1, \quad i \in Q(m) \tag{2.4}$$

$$x_0^{m,1} = 0, \qquad m = 1, \ldots, M, \tag{2.5}$$

$$x_0^{m,t} \leq \sum_{\beta=2}^{R} \sum_{j=\beta}^{R} \sum_{i \in Q(m) \cap S_j} x_i^{m,t-\beta+1}, \qquad m = 1, \ldots, M, \quad t = 2, \ldots, T \tag{2.6}$$

$$x_i^{m,t} \in \{0,1\}, \qquad m = 1, \ldots, M, \quad t = 1, \ldots, T, \quad i \in Q(m) \cup \{0\}. \tag{2.7}$$

The minimization of objective function (2.1) has the intent of reducing the total idle capacity of molds and, consequently, concrete waste in used molds. Constraints (2.2) ensure that at most one pattern shall be assigned to mold m at period t, with the possibility of this pattern being the empty one, P_0. Constraint set (2.3) requires that all demands be satisfied. Constraints (2.4) force that, if pattern P_i is initiated in mold m at period t, then pattern P_0 is associated to mold m in the next $E_i - 1$ periods. Note that the right-hand side of the constraint remains unconstrained, in case no patterns is associated to mold m at period t. Constraints (2.5) force that pattern P_0 is not associated to any mold at the first period. Constraint set (2.6) establish that P_0 shall only be used in the m-th mold if there is some pattern P_i associated with a previous period

in the same mold, such that P_i's production has not yet been completed. Constraints (2.7) define the domain of the decision variables.

2.3.2 Model for minimizing the makespan

In model (M1), minimizing the objective function (2.1) could cause some molds to be unnecessarily filled, particularly in the final periods of production, since the goal was to reduce waste. The next model switches the focus from waste to the time required to fulfill the demand. It makes use of an alternative objective function that captures the number of uninterrupted periods during which at least one mold is used before the production of all the demanded beams is completed. This corresponds to the criterion often used in scheduling problems that measures the time of completion of all jobs, or makespan.

In order to express the minimization of makespan in the model, we introduce another type of decision variable associated with the fact that there is at least one mold used at period t:

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

The following model is a specialization of model (M1) and requires the minimization of the makespan:

(M2) minimize

$$\sum_{t=1}^{T} z_t \tag{2.8}$$

subject to

$$(2.2) - (2.7)$$

$$M\, z_t \geq \sum_{m=1}^{M} \left(\sum_{i \in Q^*(m)} x_i^{m,t} \right), \qquad t = 1, \ldots, T \tag{2.9}$$

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

$$t = 1, \ldots, T-1 \tag{2.10}$$

$$z_t \in \{0, 1\}, \qquad t = 1, \ldots, T. \tag{2.11}$$

Model (M2) includes two additional sets of constraints. Constraint set (2.9) ensures that $z_t = 1$ when period t is used for the production of any beam, for each period t. Note that the

periods in which only P_0 is used are taken into account by the correspond constraint from (2.9), since those patterns correspond to actual production of beams. If no pattern is assigned to any mold during period t, then the corresponding variable z_t is not constrained. Since model (M2) minimizes (2.8) then z_t will be set to zero whenever (2.9) does not impose $z_t = 1$.

Minimizing (2.8) subject to (2.2)-(2.7) and (2.9)- (2.11) effectively minimizes the number of days in which production takes place. However, a solution satisfying those constraints might still involve periods of inactivity (i.e., periods in which the production is interrupted), followed by periods of activity. This means that the time of production of the latest beam produced is not necessarily as early as possible. In order to properly capture the makespan of the production plan and accomplish its minimization, we use constraint set (2.10): once the production is interrupted in the m-th mold at period t, it never resumes. Thus, the complete model minimizes the number of days in which production takes place, while ensuring that those days are contiguous.

A desirable property of model (M2) in the scenario of continuous production is that, once a mold becomes idle, it can be used to start the production of beams to satisfy a demand that was not yet available during the scheduling of the current production plan.

Model (M2) also can be formulated in an alternative way using variable z as an integer variable that defines the makespan in model (aM2):

(aM2) minimize

$$z \tag{2.12}$$

subject to

$$(2.2) - (2.7)$$

$$z \geq t \sum_{i \in Q^*(m)} x_i^{m,t}, \qquad t = 1, \ldots, T,$$

$$m = 1, \ldots, M \tag{2.13}$$

$$z \in \{1, \ldots, T\}. \tag{2.14}$$

Constraints (2.13) with objective function (2.12) minimization state that z is equal to the index of the last period used to produce beams.

2.3.3 Model for minimizing the total completion time

Although model (M2) generates solutions with less molds unnecessarily filled than model (M1), it still might return solutions with a great amount of unnecessary beams, i.e. more beams than the demand, which have to be stocked. Then, one way of avoiding unnecessary beams is minimizing total completion time to achieve the demand. In order to do that, we define model (M3) as follows:

(M3) minimize

$$\sum_{t=1}^{T}\sum_{m=1}^{M}\Big(\sum_{i\in Q^*(m)} x_i^{m,t}\Big) \tag{2.15}$$

subject to

$$(2.2)-(2.7),\ (2.10).$$

Objective function (2.15) aims at minimizing the total completion time of beam production.

2.3.4 Overview of the models

Model (M1) involves $\mathcal{O}(MTr)$ decision variables and $\mathcal{O}(q+MTr)$ constraints, with $q=\sum_{c=1}^{C} q_c$, while model (M2), (aM2) and (M3) have $\mathcal{O}(MTr)$ variables and $\mathcal{O}(q+MTr)$ constraints, as well. We shall not prove this bound formally, as it relies on a straightforward counting argument based on the indices of variables and constraint in models (M1), (M2), (aM2), and (M3). Depending on the total number of possible patterns, there may be an excessive number of variables in all models. In a practical scenario, the unavailability of certain lengths for given beam types might limit the quantity of patterns. Differently, this number can be limited by the exclusive usage of sets of patterns with specific properties. In Subsections 2.3.5 and 2.3.6, we discuss restrictions on the type of patterns used and why restricting the models in such ways are reasonable approaches.

Models (M1), (M2), and (M3) are linear and have only binary decision variables, a fact that allows for their solution via standard integer linear programming software. A disadvantage of model (aM2) is that it includes one general integer variable. It is interesting to note that all models are also amenable to solution via an iterative scheme of column generation, in

which the set of patterns available are generated on demand. This might prove to be useful when dealing with problems that admit very large numbers of patterns.

2.3.5 Maximal patterns

A pattern $P_i = (c, (a_1, a_2, \ldots, a_{q_c}))$ is defined as *maximal* with respect to the m-th mold if it is not possible to add any beam of type c to P_i without violating the capacity of the mold. In our models, we only used variables associated with maximal patterns in their respective molds: that is, a variable $x_i^{m,t}$ will only exist in the model if P_i is a maximal pattern in mold m.

After excluding variables that are associated with non-maximal patterns, there is typically a substantial reduction in the number of variables of all models. This, in turn, can improve their solution times considerably. However, solutions with maximal patterns may produce beam surplus as compared to non-maximal patterns, leading to an increase of the stock size.

It is important to remark that, by excluding those variables, no optimal solution of model (M1) is lost, as the following theorem shows.

Theorem 2.3.2. *Restricting model (M1) to use only maximal patterns does not modify its sets of optimal solutions.*

Proof. Let us first note that a pattern $P_i = \left(c, (a_1, a_2 \ldots, a_{q_c})\right)$ that is not maximal with respect to the m-th mold can always be transformed into a maximal pattern with respect to that mold via the addition of more beams of its own type.

Without loss of generality, let us assume that $l(c,1) < l(c,2) < \cdots < l(c,q_c)$ holds for each beam type c. Let $\mathscr{P}$ be the set of all patterns, and let us define $I : \mathscr{P} \to \mathscr{P}$, such that

$$I : \left(c, \left(a_1, a_2, \ldots, a_{q_c}\right)\right) \mapsto \left(c, \left(a_1 + \delta, a_2, \ldots, a_{q_c}\right)\right),$$

with

$$\delta = \left\lfloor \frac{L_m - \sum_{k=1}^{q_c} l(c,k) a_k}{l(c,1)} \right\rfloor .$$

If P_i is not maximal with respect to the m-th mold, then we have $\delta \geq 1$ and, therefore, $u(I(P_i)) > u(P_i)$. Moreover, $I(P_i)$ is maximal with respect to the m-th mold: the leftover capacity in $I(P_i)$ is not enough to support the addition of yet another beam to the pattern.

Let us assume, for the sake of contradiction, that $\hat{x}$ is an optimal solution of model (M1) that uses a non-maximal pattern, i.e., $x_s^{m,t} = 1$, for some mold m, period t and pattern P_s that is non-maximal with respect to m. Now, let $\bar{x}$ be a solution obtained as follows:

$$\bar{x}_\ell^{m,t} = \begin{cases} 0, & \text{if } \ell = s; \\ 1, & \text{if } P_\ell = I(P_s); \\ \hat{x}_\ell^{m,t}, & \text{otherwise,} \end{cases}$$

for all $t = 1, \ldots, T$. In other words, whenever pattern P_s is used in the m-th mold, we replace it with $P_j = I(P_s)$, the maximal pattern (with respect to mold m) obtained as shown above.

Note that $\bar{x}$ is a feasible solution to model (M1). Indeed, constraints (2.2) are satisfied because we only set $\bar{x}_j^{m,t} = 1$ when the corresponding variable $\bar{x}_s^{m,t}$ was set to 0, thereby maintaining the left-hand side of (2.2) unchanged with respect to $\hat{x}$. Additionally, (2.3) is satisfied because $P_j(c,k) \geq P_s(c,k)$, for all k. Constraints (2.4) are satisfied because if $\hat{x}_s^{m,t} = 1$, then $\bar{x}_j^{m,t} = 1$ forces the same set of $x_0^{m,t+\alpha}$ variables to be set to 1. Finally, (2.6) is satisfied because patterns P_s and P_j have the same curing time.

If we denote by $v(\cdot)$ the objective function value of a solution to (M1), we can write

$$v(\bar{x}) = v(\hat{x}) - F_s^m \sum_{t=1}^{T} \hat{x}_s^{m,t} + F_j^m \sum_{t=1}^{T} \bar{x}_j^{m,t}(1 - \hat{x}_j^{m,t}),$$

where the first summation concerns the periods in which pattern P_s is used in mold m in solution $\hat{x}$, but not in $\bar{x}$, while the second summation only takes into account the periods in which pattern P_j is used in mold m in solution $\bar{x}$ but not in $\hat{x}$.

Now, since $F_s^m = E_s(L_m - u(P_s))$, $F_j^m = E_j(L_m - u(P_j))$, $E_s = E_j$, and $u(P_j) > u(P_s)$, we have $F_s^m > F_j^m$. Moreover, $\sum_{t=1}^{T} \hat{x}_s^{m,t} = \sum_{t=1}^{T} \bar{x}_j^{m,t}(1 - \hat{x}_j^{m,t})$ by construction of $\bar{x}$. Therefore, $v(\bar{x}) < v(\hat{x})$, contradicting the optimality of $\hat{x}$. Thus, since $\hat{x}$ cannot use a non-maximal pattern, we have shown that restricting (M1) to use only maximal patterns does not change its set of optimal solutions. □

In fact, we have just shown the following result:

Corollary 2.3.3. *An optimal solution of model (M1) only uses maximal patterns.*

A result that is similar to Theorem 2.3.2 holds for model (M2). We state it here without proof, since the argument is very similar to the one used in the proof of Theorem 2.3.2.

Theorem 2.3.4. *Restricting model (M2) and (aM2) to using only maximal patterns does not modify their sets of optimal solutions.*

Indeed, replacing a non-maximal pattern with a maximal one in an optimal solution to model (M2) or (aM2) will not have an impact on the makespan because the actual number of days used to fulfill the demand will remain unaffected, given that all patterns of a given type have the same curing time.

Theorem 2.3.5. *Restricting model (M3) to using only maximal patterns does not modify its sets of optimal solutions.*

As in the case of model (M2), we can see that an optimal solution value of the total completion time will not be affected after replacing a non-maximal pattern with a maximal one. Thus, we can restrict models (M1), (M2), (aM2) and (M3) to maximal patterns without affecting the optimal values of their objective functions (2.1), (2.8), (2.12) and (2.15).

2.3.6 Size-reduction heuristic

Size-reduction heuristics are solution methods that consist in solving a reduced version of the MILP model in which only a subset of variables is considered. This means that we can drastically reduce the size of the MILP model and depending on the choice of the subset we may be led to promising sub-optimal solutions in shorter execution times and less memory usage. To cite some examples, (FANJUL-PEYRO; RUIZ, 2011) proposed several size-reduction heuristics for the unrelated parallel machines scheduling problem, reducing the number of machines to only a subset of promising ones taking several criteria into consideration. (FANJUL-PEYRO *et al.*, 2017) introduced some matheuristics to the unrelated parallel machines scheduling problem with additional resources. One of which consists in a size-reduction method named as *job-machine reduction*. Such method involves selecting only variables in which the jobs are associated to the ℓ "best" machines, otherwise they are removed from the MILP model.

Regarding the HPPBMPP, since the number of maximal patterns still can be too large for state-of-the-art solvers to handle the corresponding MILP model, we can select a subset of maximal patterns to solve the problem, thus not necessarily leading us to an optimal solution, or even a feasible one, for the global problem. Since the number of patterns in the problem is smaller, the number variables and constraints in the MILP model will be smaller.

We define q_c*-maximal patterns* as a subset of patterns that are maximal on the shortest mold from an specific instance that covers the largest number of distinct lengths of its beam type. For example, a set q_c-maximal patterns in which $q_c = 2$ is a set that has patterns

that contain at least 2 beams of distinct lengths. If there is no pattern that covers all beam lengths of a certain type, the set q_c-maximal patterns will be composed of by patterns that covers $q_c - 1$ distinct beam lengths, and so on until the set of patterns covers each beam length. Since one characteristic of the problem in practice is that usually there are molds large enough to accommodate a large quantity of beams, it is highly expected that there are patterns that covers all q_c beam lengths for each beam type.

Table 1 – Instance hbp1_30_1

Number of beam types		1					
Number of molds		30					
Number of periods		2					
Molds length (m)		60					
Type 1	Cure time	1					
	Number of beams	6					
	Lengths (m)	1.15	3.10	3.20	3.80	5.60	5.70
	Demands	26	26	27	13	20	22

The sets of patterns, maximal patterns, and q_c-maximal patterns generated for instance hbp1_30_1, have cardinalities 128674, 12078, and 1732, respectively. A large quantity of patterns is discarded when we consider only maximal patterns or q_c-maximal patterns. The number of maximal patterns and q_c-maximal patterns are equivalent to 9.39% and 1.35% of all patterns, respectively. The methods to generate such patterns are described afterwards, in Section 2.5.

2.4 Priority rules

In this section we propose six constructive heuristics, which we refer to as *priority rules*, to obtain feasible solutions for the problems under study. Each priority rule that we propose consists in, whenever a mold is freed, selecting a beam type, according to some priority measure regarding the curing time, whose demand has not been attended, and associating it to the current freed mold. Then, we fulfill the current mold with beams of the selected beam type following a second priority measure regarding beam lengths until the demand of the current beam type is achieved or the pattern associated to the current mold is maximal in such mold.

Note that each heuristic described in this section will return solutions that satisfy the demand with no beam surplus, which may lead us to solutions that are composed of patterns that are not necessarily maximal in their respective molds. Regarding this, each of the priority

rules that we propose have two phases: the first phase consists in generating a solution producing all demanded beams; the second phase consists in converting each of the patterns used in such solution into maximal patterns. This phase involves filling the patterns with beams of its type from the largest one to the shortest one in matter of length, until each of the generated pattern is maximal in its respective mold.

The priority measures proposed for curing time are:

- Shortest curing time first: consists in selecting the beam type with the shortest curing time first among beam types that did not achieved their respective demands;
- Longest curing time first: consists in selecting the beam type with the longest curing time first among beam types that did not achieved their respective demands.

The priority measures proposed for beam lengths are:

- Shortest length first: to select the beam with the shortest length first among beam length from a given type whose demands have not yet been achieved;
- Largest length first: to select the beam with the largest length first among beam length from a given type whose demands have not yet been achieved;
- Alternate lengths: to select alternately the beam with the shortest length and the beam with the longest length whose demands have not yet been achieved;

Based on the measures described above, we name the proposed priority rules as follows:

- Shortest curing time shortest length first (SCTSL);
- Shortest curing time largest length first (SCTLL);
- Shortest curing time alternate length first (SCTAL);
- Longest curing time shortest length first (LCTSL);
- Longest curing time largest length first (LCTLL);
- Longest curing time alternate length first (LCTAL).

In Table 2 we describe on which priority measure the priority rules are based.

Table 2 – Description of priority rules proposed

Heuristic	Priority measure	
	curing time	Beam length
SCTSL	Shortest	Shortest
SCTLL	Shortest	Largest
SCTAL	Shortest	Alternate
LCTSL	Longest	Shortest
LCTLL	Longest	Largest
LCTAL	Longest	Alternate

The complexity of the proposed priority rules is polynomial and it is given by $\mathscr{O}\Big(C\log C+\sum_{c=1}^{C} q_c \log q_c + M\sum_{c=1}^{C}\sum_{k=1}^{q_c} d(c,k) + MT\sum_{c=1}^{C} q_c\Big)$.

2.5 Computational tests

In this section, we evaluate the performance of the models proposed in Section 2.3 on a set of benchmark instances. We introduce a set of instances that are based on data arising from a real-life scenario. The different instances represent a sample of the variability of the problem's parameters, such as demand, number of molds, and mold lengths. We also discuss some practical aspects of the implementation of the four models.

2.5.1 Pattern generation

Instead of carrying out an exhaustive enumeration, we generated the desirable patterns for a given instance using a constraint programming model. Consider the following notation:

- K: the largest number of different lengths among beam types, i.e. $K = \max\{q_c : c = 1,\ldots,C\}$. 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 $K = 6$.
- L: the largest capacity among the molds, i.e. $L = \max\{L_m : m = 1,\ldots,M\}$.
- $V \in \{1,\ldots,C\}$ is a decision variable that corresponds to the type of beam used by the pattern.
- $A \in \mathbb{Z}^K$: a vector of decision variables, with A_j representing the number of beams of the length ℓ_j, for all $j \in \{1,\ldots,K\}$. Note that, for a type c with $q_c < k$, the components $A_{q_c+1},\ldots,A_k$ are necessarily zero. Given a pattern P_i of type c, the possibly nonzero components of vector A correspond to $[\mathscr{N}_i(c,j)]_{j=1}^{q_c}$.

- $P = P(V,A) = \big(V, (A_1, \ldots, A_{qV})\big)$: the generated pattern.
- ε: a parameter ($0 < \varepsilon < L$) that establishes a tolerance for avoiding the generation of an empty pattern, or patterns with very low capacity.

We present the model as follows:

$$1 \leq V \leq C, \tag{2.16}$$

$$A_j = 0, \qquad \text{if } V = c, \qquad c = 1, \ldots, C,\ j = q_c + 1, \ldots K, \tag{2.17}$$

$$\varepsilon < \sum_{j=1}^{q_c} l(c,j) \cdot A_j \leq L, \qquad \text{if } V = c, \qquad c = 1, \ldots, C. \tag{2.18}$$

Constraint (2.16) defines the domain of the decision variable V. Constraint set (2.17) implies that if the generated pattern is of type c then it includes no beam of size $l(c, j)$, such that $j > q_c$. Constraint set (2.18) imposes that the capacity used by the generated pattern is simultaneously larger than ε and no larger than the length of the longest mold. The empty pattern is, therefore, not generated and has to be manually included in the final set of patterns. We utilized the Gecode solver (SCHULTE *et al.*, 2016) to enumerate all the solutions of model (2.16) - (2.18).

Finally, due to the size of certain instances, we decided to generate only patterns satisfying:

$$\varepsilon = \alpha - \omega, \tag{2.19}$$

with α equal to the capacity of the shortest mold and ω equals the length of the longest beam among all the beam types.

The pattern generator is flexible and can be conveniently calibrated to restrict the model to use only a certain subset of patterns that meet specific criteria of the decision maker. The use of patterns with a maximum number of beams to be cut (for instance, a maximum of 5 pieces in each pattern) is an example of such a criterion.

2.5.2 Instance generation

The instances used in this chapter were randomly generated based on an existing order arising from a real-world production plant. For privacy reasons, we are not allowed to use or provide here the actual data coming from the aforementioned instance.

We implemented an instance generator using the MATLAB programming language, with parameters that allow us to vary the number of beams, the number of molds, the maximum value for the demands, as well as the number of distinct curing times, and their maximum value.

In our experiments, the mold length is fixed at 60 and the possible beam lengths are: 1.15, 2.5, 2.9, 3.05, 3.1, 3.2, 3.65, 3.8, 3.95, 4.05, 4.35, 4.6, 5.05, 5.6, 5.7, 5.95, 6, 6.45, 6.65, 6.9 and 7.15. From this set, we uniformly select a sample of 7 lengths for each type of beam and afterwards we remove the duplicate lengths. The demand of each beam length, for each type, is independently and uniformly selected from the set $\{12,\ldots,35\}$.

The value of T is initially calculated for each instance in the following way:

$$T = \left\lceil \frac{\sum_{i=1}^{C} t_c \cdot \left(\sum_{k=1}^{q_c} l(c,k) \cdot d(c,k) \right)}{\sum_{m=1}^{M} L_m} \right\rceil. \tag{2.20}$$

Considering that the calculation of T this way is a lower bound on the maximum number of periods necessary for the total production, the problem may well become infeasible. For this reason, if T was less than or equal to 10, we increased its value in one unit; otherwise, we increased the value of T by 10 percent. We avoided the calculation of a guaranteed upper bound, because in case such a bound was not particularly tight, its value could significantly increase the number of decision variables in our models, hindering their computational solution via standard software for integer linear programming.

2.5.3 Experimental evaluation

All computational tests carried out in this subsection were performed on an Intel (R) Core i5-3470 CPU @ 2 x 3.20GHz processor, with 8 GB RAM, running 64-bit Windows 7 Professional. The software used for implementing the models were Gecode 5.0.0 (for pattern generation) and IBM ILOG CPLEX Optimization Studio 12.7.1, using the C++ programming language with IDE Microsoft Visual Studio 2015 (for solving the integer linear programming models). Note that constraints (2.2) can be implemented in Concert technology as SOS1 constraints. Carrying out preliminary tests we concluded that there were no significant gains on such implementation for our problem as compared to the linear formulation. Therefore we decided to implement such constraints as classical linear constraints in every experimental evaluation described in this work.

For the computational tests in this subsection we used four groups of instances, each containing 10 instances, classified as follows:

1. Instances with 1 beam type, with $E_1 = 1$;
2. Instances with 2 beam types, with $E_1 = 1$ and $E_2 = 2$;
3. Instances with 3 beam types, with $E_1 = 1$, $E_2 = 2$ and $E_3 = 3$;
4. Instances with 4 beam types, with $E_1 = 1$, $E_2 = 2$, $E_3 = 3$ and $E_4 = 4$;

Instances of group 1 have names starting with "hbp1", and instances of groups 2, 3, and 4 are named "hbp2", "hbp3", and "hbp4", respectively. Each group is divided into two subgroups, each one containing 5 instances. The first subgroup contains instances with 15 molds while the second includes instances with 30 molds. The full name of each instance reflects its group, number of molds, and a sequential number (from 1 to 10) that identifies its subgroup.

All models were solved using only variables corresponding to maximal patterns in their respective molds. The results obtained with each model are presented in terms of best value of the objective function and total running time in seconds, for each instance. The number of molds, the maximum number of production periods, and the number of patterns for each instance are also shown in the table. The time limit used for solving each instance with CPLEX was 3,600 seconds. We did not impose a limit on the time taken to generate patterns with Gecode, since this preprocessing phase requires very short times as compared to solving the integer programming model. The running times shown in the tables correspond to the sum of the CPLEX running times.

We define the notation used in the test tables in this subsection as follows:

- LPv: Value of objective function of the linear relaxation problem;
- LPt: Running time of linear relaxation problem in seconds;
- IPv: Value of objective function of the integer linear problem;
- IPt: Running time of the integer linear problem in seconds;
- Gap: Gap between the best integer objective and the objective of the best node remaining (0% means an optimal solution);
- BCn: Number of nodes in the branch-and-cut tree used by CPLEX.

The objective function values corresponding to the entries with "3,600" in the "IPt" columns are not necessarily optimal. In those cases where the value of gap is nonzero, the instance was not solved to optimality within the prescribed maximum solution time. For this reason, the objective function values reflect the best solution found within 3,600 seconds of

running time.

Some instances were not solved by some of the models. Those cases are marked with "–". Two possible scenarios were documented: (i) the amount of memory available was not enough to support CPLEX's solution procedure with the default parameters, and the optimization process was interrupted before finding a feasible solution; or (ii) the time limit of 3,600 seconds was exceed before a feasible solution was found. It is important to note that patterns were successfully generated for all instances in our tests. Thus, any memory-related interruption occurred as part of CPLEX's solution process.

The number of patterns vary widely, and there seems to be no directly relation between that number and M or T, although the larger the molds are the more patterns we have since there are more combinations of beam lengths that can fit into the molds. The number of patterns can also be affected by the randomly chosen lengths of beams.

We can see in Table 3, for each instance, the number of molds in column M, the number of time periods in column T, the number of patterns generated that are maximal in molds of length 60 plus pattern P_0 in column Maximal patterns, and the number of patterns which covers all lengths for their type that are maximal in molds of length 60 plus pattern P_0 in column q_c-maximal patterns.

The means of number of patterns generated for type 1, type 2, type 3 and type 4 are 4451.30, 8734.60, 14925.70 and 15979.70, respectively, and the standard deviations are 4451.30, 8734.60, 14925.70 and 15979.70. We can see that the number of patterns reduction when we adopt exclusively q_c-maximal patterns is enormous, with an average reduction of 84.45%. Thus we can see that, for the instances that we generated for this work, the more types we have the greater will be the number of patterns, but high standard deviations lead us to deduce that not only the number of types influences the number of patterns, but the number of beams for each, as well as their lengths (which are randomly generated).

2.5.3.1 Computational tests with maximal patterns

In these experiments we carry out computational tests for the models (M1), (M2), (aM2) and (M3) proposed in this chapter considering all maximal patterns plus pattern P_0 for the generated instances.

In Table 4 we can see that almost all instances were solved to optimality within the time limit of 3,600 seconds. With instance hbp4_30_3 we could only obtain the linear relaxation

Table 3 – Instance information

Instance	T	M	Maximal patterns	q_c-Maximal patterns	Patterns set reduction
hbp1_15_1	2	15	3,348	831	75.18%
hbp1_15_2	2	15	7,944	762	90.41%
hbp1_15_3	2	15	1,322	235	82.22%
hbp1_15_4	2	15	256	83	67.58%
hbp1_15_5	2	15	7,200	640	91.11%
hbp1_30_1	2	30	12,078	1,732	85.66%
hbp1_30_2	2	30	420	162	61.43%
hbp1_30_3	2	30	1,452	254	82.51%
hbp1_30_4	2	30	4,365	392	91.02%
hbp1_30_5	2	30	6,128	1,953	68.13%
hbp2_15_1	4	15	6,595	386	94.15%
hbp2_15_2	4	15	13,389	1,281	90.43%
hbp2_15_3	4	15	7,867	669	91.50%
hbp2_15_4	4	15	4,287	333	92.23%
hbp2_15_5	3	15	3,638	743	79.58%
hbp2_30_1	2	30	15,588	1,693	89.14%
hbp2_30_2	3	30	6,024	578	90.41%
hbp2_30_3	3	30	5,211	604	88.41%
hbp2_30_4	2	30	20,192	3,190	84.20%
hbp2_30_5	2	30	4,555	1,103	75.78%
hbp3_15_1	6	15	14,771	1,489	89.92%
hbp3_15_2	4	15	32,897	6,309	80.82%
hbp3_15_3	5	15	15,049	1,587	89.45%
hbp3_15_4	5	15	17,221	2,207	87.18%
hbp3_15_5	5	15	6,788	1,066	84.30%
hbp3_30_1	4	30	28,874	3,468	87.99%
hbp3_30_2	4	30	9,798	1,343	86.29%
hbp3_30_3	3	30	8,744	1,277	85.40%
hbp3_30_4	3	30	10,723	1,287	88.00%
hbp3_30_5	3	30	4,392	1,023	76.71%
hbp4_15_1	7	15	19,271	4,368	77.33%
hbp4_15_2	8	15	12,150	1,573	87.05%
hbp4_15_3	7	15	7,528	1,410	81.27%
hbp4_15_4	8	15	15,007	1,791	88.07%
hbp4_15_5	8	15	7,745	859	88.91%
hbp4_30_1	5	30	6,706	823	87.73%
hbp4_30_2	6	30	12,063	1,124	90.68%
hbp4_30_3	4	30	41,005	6,475	84.21%
hbp4_30_4	5	30	23,252	3,219	86.16%
hbp4_30_5	4	30	15,070	3,073	79.61%

solution. Note that the only integer solution found with a value different than zero was with instance hbp3_30_4, what can be explained by the fact that since the molds are much bigger than the beam lengths and usually there are many different beam lengths, this situation can

result in many combinations between them leading to a high probability to have patterns with an associated waste zero. It is also interesting to observe that only instance hbp4_15_3 was not solved in the root of the branch-and-cut tree by the solver. We can also see that there is a trend that the more patterns there are in an instance the more difficult it is to solve it.

In Table 5 we can see that 12 instances could not be solved to optimality within the time limit of 3,600 seconds by model (M2); 3 of them could not even be solved to an integer solution. The largest gap among the instances was 100% and for the vast majority of instances CPLEX did not use more than the root node in the branch-and-cut procedure. Also in Table 5 we can see that 15 instances could not be solved to optimality within the time limit of 3,600 seconds by model (aM2), 14 of them could not even be solved to an integer solution. No integer solutions were found for none of instances of subgroup 4. These results led us to infer that model (M2) is better for solving the problem considering makespan minimization for this group of instances than model (aM2).

In Table 6 we can see that 28 instances could not be solved to optimality within the time limit of 3,600 seconds by model (M3) and only 2 of them could not even be solved to a feasible integer solution. The largest gap was 56.52% with instance hbp4_15_5. Unlike the other models, for the majority of instances the solver used more branch-and-cut nodes than only the root.

The general outlook revealed by the experiments suggests that all models can be used for the exact solution of instances with a few types of beams and a limited number of patterns. However, as those parameters increase, solving the models can become quite difficult. The size of the models increases relatively fast with the parameters, demanding the use of substantially more memory, or slowing down the solution process in a significant way.

2.5.3.2 Computational tests with size-reduction heuristic

In these experiments we carry out computational tests for the models (M1), (M2) and (M3) using the size reduction approach (SRH) proposed in this chapter, that consists in considering only q_c-maximal patterns plus pattern P_0 on the group of generated instances. We observe the behavior of solutions using this method and compare to the models considering all maximal patterns plus pattern P_0.

We can see in Table 7 that from the 39 instances solved to optimality with all maximal patterns by model (M1) we got the optimal solution for the 25 instances with SRH. It is important

Table 4 – Results of computational tests with model (M1)

Instance	M1					
	LPv	LPt	IPv	BCn	Gap	IPt
hbp1_15_1	0.00	1.21	0.00	0	0.00%	2.22
hbp1_15_2	0.00	2.97	0.00	0	0.00%	5.34
hbp1_15_3	0.00	0.39	0.00	0	0.00%	0.70
hbp1_15_4	0.00	0.09	0.00	0	0.00%	0.17
hbp1_15_5	0.00	2.25	0.00	0	0.00%	5.06
hbp1_30_1	0.00	7.88	0.00	0	0.00%	15.50
hbp1_30_2	0.00	0.25	0.00	0	0.00%	0.47
hbp1_30_3	0.00	0.85	0.00	0	0.00%	1.50
hbp1_30_4	0.00	2.85	0.00	0	0.00%	6.39
hbp1_30_5	0.00	3.77	0.00	0	0.00%	8.68
hbp2_15_1	0.00	7.97	0.00	0	0.00%	14.04
hbp2_15_2	0.00	30.14	0.00	0	0.00%	39.86
hbp2_15_3	0.00	9.08	0.00	0	0.00%	22.90
hbp2_15_4	0.00	4.20	0.00	0	0.00%	8.83
hbp2_15_5	0.00	2.08	0.00	0	0.00%	6.67
hbp2_30_1	0.00	12.48	0.00	0	0.00%	28.20
hbp2_30_2	0.00	6.95	0.00	0	0.00%	17.03
hbp2_30_3	0.00	7.12	0.00	0	0.00%	12.29
hbp2_30_4	0.00	23.25	0.00	0	0.00%	48.95
hbp2_30_5	0.00	3.13	0.00	0	0.00%	8.83
hbp3_15_1	0.00	23.27	0.00	0	0.00%	194.30
hbp3_15_2	0.00	96.69	0.00	0	0.00%	292.76
hbp3_15_3	0.00	45.72	0.00	0	0.00%	126.93
hbp3_15_4	0.00	52.12	0.00	0	0.00%	119.55
hbp3_15_5	0.00	9.51	0.00	0	0.00%	30.44
hbp3_30_1	0.00	85.37	0.00	0	0.00%	714.98
hbp3_30_2	0.00	17.73	0.00	0	0.00%	54.53
hbp3_30_3	0.00	11.15	0.00	0	0.00%	34.60
hbp3_30_4	2.55	25.34	2.55	0	0.00%	49.84
hbp3_30_5	0.00	4.24	0.00	0	0.00%	15.23
hbp4_15_1	0.00	87.60	0.00	0	0.00%	1035.95
hbp4_15_2	0.00	75.15	0.00	0	0.00%	206.15
hbp4_15_3	0.00	11.78	0.00	1312	0.00%	532.82
hbp4_15_4	0.00	34.48	0.00	0	0.00%	313.79
hbp4_15_5	0.00	18.60	0.00	0	0.00%	135.39
hbp4_30_1	0.00	16.97	0.00	0	0.00%	70.77
hbp4_30_2	0.00	51.36	0.00	0	0.00%	266.83
hbp4_30_3	0.00	577.39	–	–	–	–
hbp4_30_4	0.00	320.97	0.00	0	0.00%	1556.48
hbp4_30_5	0.00	26.37	0.00	0	0.00%	108.52

to see that we were also able to find the optimal solution for instance hbp4_30_3, the instance for which we could not even find a feasible integer solution solving by model (M1) with all maximal patterns. We can infer that such a solution is a global optimum for the whole problem since the

Table 5 – Results of computational tests with models (M2) and (aM2)

Instance	M2						aM2					
	LPv	LPt	IPv	BCn	Gap	IPt (s)	LPv	LPt	IPv	BCn	Gap	IPt (s)
hbp1_15_1	0.44	1.74	1	0	0.00%	3.29	0.29	1.01	1	0	0.00%	3.07
hbp1_15_2	0.78	3.24	1	0	0.00%	8.25	0.52	2.70	1	0	0.00%	12.75
hbp1_15_3	0.66	0.49	1	0	0.00%	2.09	0.44	0.41	1	0	0.00%	1.13
hbp1_15_4	0.65	0.13	1	0	0.00%	0.36	0.43	0.15	1	0	0.00%	0.23
hbp1_15_5	0.90	2.84	1	0	0.00%	10.50	0.60	2.35	1	0	0.00%	12.63
hbp1_30_1	0.27	10.63	1	0	0.00%	32.57	0.18	9.02	1	0	0.00%	35.26
hbp1_30_2	0.24	0.30	1	0	0.00%	0.67	0.16	0.28	1	0	0.00%	0.71
hbp1_30_3	0.34	1.09	1	0	0.00%	2.63	0.23	0.85	1	0	0.00%	2.83
hbp1_30_4	0.33	3.58	1	0	0.00%	8.75	0.22	2.88	1	0	0.00%	10.26
hbp1_30_5	0.21	5.02	1	0	0.00%	13.72	0.14	4.12	1	0	0.00%	16.47
hbp2_15_1	1.68	8.74	3	111	0.00%	51.62	0.81	7.85	3	0	0.00%	57.95
hbp2_15_2	1.69	18.88	3	0	0.00%	89.72	0.81	33.87	3	0	0.00%	101.65
hbp2_15_3	1.77	9.15	3	0	0.00%	36.89	0.85	10.41	3	0	0.00%	65.59
hbp2_15_4	1.59	5.86	3	0	0.00%	13.48	0.76	5.93	3	0	0.00%	21.97
hbp2_15_5	1.31	2.63	2	30,333	0.00%	537.26	0.71	2.26	2	262,116	0.00%	3,306.12
hbp2_30_1	0.58	17.38	2	0	0.00%	30.12	0.39	22.30	2	0	0.00%	104.10
hbp2_30_2	0.81	9.38	2	0	0.00%	31.12	0.44	8.94	2	0	0.00%	44.50
hbp2_30_3	0.74	8.81	2	0	0.00%	23.45	0.40	8.44	2	0	0.00%	31.92
hbp2_30_4	0.66	23.10	2	0	0.00%	46.52	0.44	34.47	2	0	0.00%	108.67
hbp2_30_5	0.53	4.18	2	0	0.00%	7.44	0.35	3.90	2	0	0.00%	16.05
hbp3_15_1	2.18	51.75	6	0	50.00%	3,600.00	0.89	56.79	6	0	66.67%	3,600.00
hbp3_15_2	1.30	84.88	4	0	25.00%	3,600.00	0.62	101.92	–	–	–	–
hbp3_15_3	2.05	53.56	5	0	40.00%	3,600.00	0.90	50.31	–	–	–	–
hbp3_15_4	1.83	54.40	5	13	40.00%	3,600.00	0.80	60.61	–	–	–	–
hbp3_15_5	1.62	19.80	4	4,796	0.00%	1,068.38	0.71	27.05	4	4,759	0.00%	959.68
hbp3_30_1	1.08	157.98	3	0	0.00%	1,841.38	0.52	105.39	–	–	–	–
hbp3_30_2	1.03	30.50	3	0	0.00%	229.34	0.49	28.34	3	0	0.00%	452.44
hbp3_30_3	0.92	13.26	3	0	0.00%	26.32	0.50	13.47	3	0	0.00%	241.84
hbp3_30_4	1.04	28.60	3	0	0.00%	43.97	0.57	32.15	3	0	0.00%	440.86
hbp3_30_5	0.80	5.70	3	0	0.00%	14.60	0.43	5.52	3	0	0.00%	111.38
hbp4_15_1	2.31	96.85	7	0	42.86%	3,600.00	0.89	117.00	–	–	–	–
hbp4_15_2	2.71	61.34	–	–	–	–	1.00	56.93	–	–	–	–
hbp4_15_3	2.53	20.59	–	–	–	–	0.98	19.69	–	–	–	–
hbp4_15_4	2.61	60.55	8	0	50.00%	3,600.00	0.96	57.24	–	–	–	–
hbp4_15_5	2.55	37.39	–	–	–	–	0.94	31.79	–	–	–	–
hbp4_30_1	1.44	29.42	4	0	0.00%	633.72	0.63	28.19	–	–	–	–
hbp4_30_2	1.66	108.30	6	0	33.33%	3,600.00	0.68	91.54	–	–	–	–
hbp4_30_3	1.21	288.02	4	0	0.00%	2,092.90	0.58	652.91	–	–	–	–
hbp4_30_4	1.32	503.35	5	0	20.00%	3,600.00	0.58	334.07	–	–	–	–
hbp4_30_5	1.27	35.51	4	0	0.00%	566.33	0.61	31.96	–	–	–	–

idle capacity cannot be negative.

We can see in Table 8 that from the 31 instances that were solved to optimality by model (M2) with all maximal patterns we could find the optimal solution for 24 of them using the SRH. For the 9 instances that could not be solved to optimality by model (M2) with all maximal patterns we were able to find better objective functions values for 6 instances and for the other 3 we found equal makespans using the SRH. Also, using the SRH, we were able to obtain feasible solutions for the 3 instances that could not be solved to a feasible solution by model (M2) with all maximal patterns.

We can see in Table 9 that from the 12 instances that were solved to optimality by model (M3) with all maximal patterns we could find the optimal solution for 11 instances only

Table 6 – Results of computational tests with model (M3)

Instance	M3					
	LPv	LPt	IPv	BCn	Gap	IPt
hbp1_15_1	6.60	0.91	7	0	0.00%	2.82
hbp1_15_2	11.63	2.32	12	0	0.00%	9.41
hbp1_15_3	9.94	0.32	10	0	0.00%	2.27
hbp1_15_4	9.75	0.08	10	0	0.00%	0.24
hbp1_15_5	13.57	2.03	14	0	0.00%	8.69
hbp1_30_1	8.06	7.10	9	0	0.00%	23.79
hbp1_30_2	7.08	0.20	8	0	0.00%	0.58
hbp1_30_3	10.34	0.73	11	0	0.00%	2.06
hbp1_30_4	9.84	2.40	10	885	0.00%	62.95
hbp1_30_5	6.22	3.35	7	0	0.00%	3,600.00
hbp2_15_1	25.19	6.54	40	0	0.00%	24.06
hbp2_15_2	25.29	13.92	40	33,610	2.73%	3,600.00
hbp2_15_3	26.49	7.13	42	84,398	3.04%	3,600.00
hbp2_15_4	23.81	5.12	39	161,369	4.96%	3,600.00
hbp2_15_5	19.66	2.04	30	0	0.00%	12.40
hbp2_30_1	17.51	13.61	28	21,089	8.18%	3,600.00
hbp2_30_2	24.21	7.35	38	49,235	4.58%	3,600.00
hbp2_30_3	22.10	7.41	37	71,651	3.65%	3,600.00
hbp2_30_4	19.67	18.87	30	27,081	6.72%	3,600.00
hbp2_30_5	15.94	3.36	27	97,008	7.60%	3,600.00
hbp3_15_1	32.66	43.82	67	186	35.23%	3,600.00
hbp3_15_2	19.50	70.84	46	176	46.28%	3,600.00
hbp3_15_3	30.82	52.57	64	579	32.03%	3,600.00
hbp3_15_4	27.41	47.65	63	611	40.85%	3,600.00
hbp3_15_5	24.23	14.30	53	1,769	21.82%	3,600.00
hbp3_30_1	32.41	113.63	71	1	36.92%	3,600.00
hbp3_30_2	30.80	22.15	67	516	40.04%	3,600.00
hbp3_30_3	27.62	11.25	60	5,654	4.51%	3,600.00
hbp3_30_4	31.19	36.95	62	26,639	4.36%	3,600.00
hbp3_30_5	23.88	4.70	50	27,622	5.06%	3,600.00
hbp4_15_1	34.65	87.65	104	0	56.04%	3,600.00
hbp4_15_2	40.68	54.61	104	126	48.81%	3,600.00
hbp4_15_3	37.94	19.46	91	429	46.35%	3,600.00
hbp4_15_4	39.10	52.42	101	5	50.89%	3,600.00
hbp4_15_5	38.29	33.30	106	0	56.52%	3,600.00
hbp4_30_1	43.06	21.85	110	270	48.91%	3,600.00
hbp4_30_2	49.91	85.16	138	51	54.03%	3,600.00
hbp4_30_3	36.18	347.03	–	–	–	3,600.00
hbp4_30_4	39.54	390.46	–	–	–	3,600.00
hbp4_30_5	38.23	27.03	94	0	48.74%	3,600.00

using the SRH. From the 26 instances that could not be solved to optimality by model (M3) with all maximal patterns we found equal or better objective function values for all them, with strictly better objective function values for 14 of them. Using the SRH, we were also able to find feasible

Table 7 – Results of computational tests with model (M1) using the SRH

Instance	M1					
	LPv	LPt	IPv	BCn	Gap	IPt (s)
hbp1_15_1	0.00	0.20	0.00	0	0.00%	0.14
hbp1_15_2	0.00	0.19	0.00	0	0.00%	0.18
hbp1_15_3	0.00	0.06	0.00	0	0.00%	0.07
hbp1_15_4	1.75	0.03	1.80	0	0.00%	0.12
hbp1_15_5	0.00	0.16	0.00	0	0.00%	0.15
hbp1_30_1	0.00	0.84	0.00	0	0.00%	0.70
hbp1_30_2	0.00	0.08	0.00	0	0.00%	0.07
hbp1_30_3	0.00	0.12	0.00	0	0.00%	0.10
hbp1_30_4	0.00	0.18	0.00	0	0.00%	0.19
hbp1_30_5	0.00	0.91	0.00	0	0.00%	0.71
hbp2_15_1	1.04	0.20	1.05	0	0.00%	1.10
hbp2_15_2	0.00	0.73	0.00	0	0.00%	3.52
hbp2_15_3	0.00	0.32	0.90	0	0.00%	2.59
hbp2_15_4	0.00	0.18	0.00	0	0.00%	0.69
hbp2_15_5	1.10	0.28	1.20	0	0.00%	1.28
hbp2_30_1	0.00	0.82	0.00	0	0.00%	3.49
hbp2_30_2	0.00	0.43	0.00	0	0.00%	1.24
hbp2_30_3	0.00	0.44	0.10	0	0.00%	1.47
hbp2_30_4	0.00	2.08	0.00	0	0.00%	5.27
hbp2_30_5	0.00	0.57	0.20	0	0.00%	2.60
hbp3_15_1	0.00	1.42	0.00	0	0.00%	12.95
hbp3_15_2	0.00	5.19	0.30	50,199	0.00%	705.52
hbp3_15_3	0.00	1.36	0.00	0	0.00%	8.94
hbp3_15_4	0.00	1.89	0.00	0	0.00%	10.59
hbp3_15_5	0.00	0.67	0.15	2,475	0.00%	7.74
hbp3_30_1	0.00	3.68	0.00	0	0.00%	17.91
hbp3_30_2	0.00	1.33	0.00	0	0.00%	6.50
hbp3_30_3	0.00	1.07	0.00	0	0.00%	5.07
hbp3_30_4	11.55	1.29	11.65	64,777	0.00%	45.42
hbp3_30_5	0.00	0.68	0.00	0	0.00%	2.39
hbp4_15_1	0.00	5.21	0.00	0	0.00%	49.84
hbp4_15_2	0.00	2.35	1.60	29,941	41.49%	3,600.00
hbp4_15_3	0.00	1.47	0.40	1,005,780	100.00%	3,600.00
hbp4_15_4	0.00	2.24	0.00	0	0.00%	18.00
hbp4_15_5	0.00	0.97	0.55	1,999,090	100.00%	3,600.00
hbp4_30_1	2.00	1.30	2.65	282,061	0.00%	355.24
hbp4_30_2	0.00	2.16	1.15	1,602,117	18.60%	3,600.00
hbp4_30_3	0.00	10.18	0.00	0	0.00%	41.00
hbp4_30_4	0.00	6.75	0.00	0	0.00%	31.31
hbp4_30_5	0.00	2.93	0.00	0	0.00%	16.23

solutions for the 2 instances that could not be solved to a feasible solution by model (M3).

Table 8 – Results of computational tests with model (M2) using the SRH

Instance	M2					
	LPv	LPt	IPv	BCn	Gap	IPt (s)
hbp1_15_1	0.44	0.24	1	0	0.00%	0.55
hbp1_15_2	0.78	0.23	1	0	0.00%	0.72
hbp1_15_3	0.66	0.08	1	0	0.00%	0.28
hbp1_15_4	0.65	0.03	1	0	0.00%	0.13
hbp1_15_5	0.90	0.20	1	0	0.00%	0.95
hbp1_30_1	0.27	1.20	1	0	0.00%	2.49
hbp1_30_2	0.24	0.09	1	0	0.00%	0.23
hbp1_30_3	0.35	0.14	1	0	0.00%	0.36
hbp1_30_4	0.33	0.26	1	0	0.00%	0.56
hbp1_30_5	0.21	1.24	1	0	0.00%	3.32
hbp2_15_1	1.71	0.54	3	0	0.00%	1.50
hbp2_15_2	1.70	1.12	3	0	0.00%	4.48
hbp2_15_3	1.83	0.47	3	0	0.00%	2.66
hbp2_15_4	1.62	0.38	3	0	0.00%	1.12
hbp2_15_5	1.32	0.38	3	1,841,543	33.33%	3,600.00
hbp2_30_1	0.59	1.21	2	0	0.00%	3.61
hbp2_30_2	0.82	0.66	2	0	0.00%	1.78
hbp2_30_3	0.76	0.64	2	0	0.00%	1.87
hbp2_30_4	0.66	2.50	2	0	0.00%	5.31
hbp2_30_5	0.54	0.71	2	0	0.00%	2.58
hbp3_15_1	2.19	2.62	5	28,252	20.00%	3,600.00
hbp3_15_2	1.31	7.57	3	781	0.00%	158.47
hbp3_15_3	2.07	2.40	5	116,809	20.00%	3,600.00
hbp3_15_4	1.84	3.27	4	2,609	0.00%	395.11
hbp3_15_5	1.63	1.36	4	0	0.00%	22.16
hbp3_30_1	1.09	6.16	3	0	0.00%	35.33
hbp3_30_2	1.03	2.50	3	0	0.00%	13.93
hbp3_30_3	0.93	1.51	3	0	0.00%	4.59
hbp3_30_4	1.06	1.55	3	0	0.00%	9.21
hbp3_30_5	0.80	0.89	3	0	0.00%	9.10
hbp4_15_1	2.32	12.74	7	15	42.86%	3,600.00
hbp4_15_2	2.75	3.70	8	3,836	50.00%	3,600.00
hbp4_15_3	2.55	2.39	6	27,395	16.67%	3,600.00
hbp4_15_4	2.62	4.22	7	9,003	14.29%	3,600.00
hbp4_15_5	2.59	1.84	7	14,909	14.29%	3,600.00
hbp4_30_1	1.45	2.07	4	0	0.00%	53.24
hbp4_30_2	1.70	3.89	5	4,843	0.00%	533.59
hbp4_30_3	1.21	16.28	4	0	0.00%	40.79
hbp4_30_4	1.34	8.91	4	0	0.00%	113.59
hbp4_30_5	1.28	4.52	4	0	0.00%	16.12

Table 9 – Results of computational tests with model (M3) using the SRH

Instance	M3					
	LPv	LPt	IPv	BCn	Gap	IPt (s)
hbp1_15_1	6.60	0.22	7	0	0.00%	0.55
hbp1_15_2	11.65	0.24	12	0	0.00%	1.05
hbp1_15_3	9.96	0.08	10	0	0.00%	0.38
hbp1_15_4	9.79	0.02	10	0	0.00%	0.12
hbp1_15_5	13.57	0.21	14	0	0.00%	0.71
hbp1_30_1	8.06	1.05	9	0	0.00%	3.04
hbp1_30_2	7.08	0.09	8	0	0.00%	0.22
hbp1_30_3	10.35	0.15	11	0	0.00%	0.46
hbp1_30_4	9.85	0.24	10	0	0.00%	1.17
hbp1_30_5	6.23	1.20	7	0	0.00%	2.40
hbp2_15_1	25.63	0.57	40	0	0.00%	1.15
hbp2_15_2	25.57	1.06	40	2,438	0.00%	44.27
hbp2_15_3	27.45	0.49	42	0	0.00%	6.08
hbp2_15_4	24.29	0.37	39	1,789,265	4.45%	3,600.00
hbp2_15_5	19.87	0.36	32	869,712	7.43%	3,600.00
hbp2_30_1	17.78	1.17	28	174,674	7.97%	3,600.00
hbp2_30_2	24.48	0.67	38	332,998	4.14%	3,600.00
hbp2_30_3	22.74	0.69	37	2,519	0.00%	87.05
hbp2_30_4	19.81	2.65	30	143,757	6.45%	3,600.00
hbp2_30_5	16.26	0.71	27	223,157	7.13%	3,600.00
hbp3_15_1	32.90	2.61	65	15,782	16.97%	3,600.00
hbp3_15_2	19.64	7.64	45	47,173	8.16%	3,600.00
hbp3_15_3	31.02	2.81	64	38,337	19.29%	3,600.00
hbp3_15_4	27.65	3.38	58	28,532	16.53%	3,600.00
hbp3_15_5	24.47	1.38	53	87,367	3.59%	3,600.00
hbp3_30_1	32.73	6.08	71	1,239	11.64%	3,600.00
hbp3_30_2	31.00	2.54	64	27,627	19.52%	3,600.00
hbp3_30_3	27.96	1.53	60	134,206	3.00%	3,600.00
hbp3_30_4	31.85	1.54	62	47,824	0.00%	1,878.17
hbp3_30_5	24.10	0.91	50	163,957	7.38%	3,600.00
hbp4_15_1	34.85	11.61	84	501	45.56%	3,600.00
hbp4_15_2	41.18	3.82	99	1,583	30.23%	3,600.00
hbp4_15_3	38.29	2.28	87	39,966	23.40%	3,600.00
hbp4_15_4	39.36	4.10	97	10,343	25.82%	3,600.00
hbp4_15_5	38.87	1.83	103	32,940	16.75%	3,600.00
hbp4_30_1	43.57	2.10	109	38,958	12.65%	3,600.00
hbp4_30_2	50.92	3.71	131	10,734	9.70%	3,600.00
hbp4_30_3	36.32	16.38	91	0	40.00%	3,600.00
hbp4_30_4	40.13	8.82	109	596	53.15%	3,600.00
hbp4_30_5	38.46	4.39	91	672	38.77%	3,600.00

In general we can see that using the SRH with only q_c-maximal patterns is a promising way of exploring the problem leading to good feasible solutions and much less running time and/or memory usage. This set of patterns could be also used as an initial set of columns in a column decomposition approach to get an optimal solution to the problem as a whole.

2.5.3.3 Comparing solutions obtained via mathematical models and priority rules

Regarding solutions obtained by priority rules in matter of idle capacities, we can see in Table 10 that all priority rules had significantly inferior solution quality compared to the ones obtained via mathematical model, even though the elapsed time to get to these solutions was insignificant, < 0.00 seconds. In none of the instances the priority rules found neither an equal nor a best value of idle capacities than those found by models (M1) or the SRH version of model (M1). Note that in the tables in this subsection *nBV* means the number of instances for which the solution method found the best objective function value, not necessarily optimal.

With respect to solutions obtained by priority rules in matter of makespan, we can see in Table 11 that some of priority rules had makespans more similar to the ones obtained via mathematical model. The heuristic LCTAL was the best priority rule for these computational tests, identifying the best makespan among solution methods for 36 instances, being inferior only to model (M2) using the SRH, which found 37 best solutions regarding the makespan.

Considering solutions obtained by priority rules in matter of total completion times, we can see in Table 12 that the priority rules often have had higher total completion time values than models (M3) and the SRH version of model (M3). The heuristics which found the best solution regarding total completion time were SCTAL and LCTAL, each identifying the best value for 19 instances. Model (M3) using SRH was by far the best solution method for this purpose, finding the best value for 38 instances. Note that for instance hbp4_30_4, the heuristic methods found better solutions than the mathematical model. However, after an analysis of variance we saw that H_0 was not rejected, thus there is no significant difference among the priority rules with *p-values* 0.99, 0.77 and 1 for idle capacities, makespans, and total completion times, respectively.

Table 10 – Idle capacities obtained by solution methods

Instance	Idle capacities							
	SCTSL	**SCTLL**	**SCTAL**	**LCTSL**	**LCTLL**	**LCTAL**	**M1**	**M1-SRH**
hbp1_15_1	7.60	4.70	7.60	7.60	4.70	7.60	0.00	0.00
hbp1_15_2	8.35	6.35	13.30	8.35	6.35	13.30	0.00	0.00
hbp1_15_3	9.05	6.55	11.95	9.05	6.55	11.95	0.00	0.00
hbp1_15_4	15.95	15.95	15.95	15.95	15.95	15.95	0.00	1.80
hbp1_15_5	20.05	20.45	19.90	20.05	20.45	19.90	0.00	0.00
hbp1_30_1	4.60	3.85	4.95	4.60	3.85	4.95	0.00	0.00
hbp1_30_2	13.60	10.40	10.35	13.60	10.40	10.35	0.00	0.00
hbp1_30_3	15.90	17.90	15.60	15.90	17.90	15.60	0.00	0.00
hbp1_30_4	6.60	8.10	7.75	6.60	8.10	7.75	0.00	0.00
hbp1_30_5	6.45	9.95	9.35	6.45	9.95	9.35	0.00	0.00
hbp2_15_1	26.05	32.70	38.90	26.05	32.70	38.90	0.00	1.05
hbp2_15_2	21.40	24.70	27.40	21.40	24.70	27.40	0.00	0.00
hbp2_15_3	46.20	45.75	51.50	46.20	45.75	51.50	0.00	0.90
hbp2_15_4	34.15	42.95	44.40	34.15	42.95	44.40	0.00	0.00
hbp2_15_5	50.10	60.20	58.30	50.10	60.20	58.30	0.00	1.20
hbp2_30_1	39.55	33.45	26.95	39.55	33.45	26.95	0.00	0.00
hbp2_30_2	46.40	39.75	39.25	46.40	39.75	39.25	0.00	0.00
hbp2_30_3	59.55	69.40	65.85	59.55	69.40	65.85	0.00	0.10
hbp2_30_4	25.30	30.85	42.95	25.30	30.85	42.95	0.00	0.00
hbp2_30_5	18.10	22.00	25.80	18.10	22.00	25.80	0.00	0.20
hbp3_15_1	63.40	62.00	76.75	63.40	62.00	76.75	0.00	0.00
hbp3_15_2	35.60	34.25	48.25	35.60	34.25	48.25	0.00	0.30
hbp3_15_3	47.95	50.35	59.65	47.95	50.35	59.65	0.00	0.00
hbp3_15_4	49.10	46.00	49.70	49.10	46.00	49.70	0.00	0.00
hbp3_15_5	59.75	55.45	61.80	59.75	55.45	61.80	0.00	0.15
hbp3_30_1	49.75	51.85	53.35	49.75	51.85	53.35	0.00	0.00
hbp3_30_2	58.05	57.70	70.40	58.05	57.70	70.40	0.00	0.00
hbp3_30_3	69.50	74.90	63.55	69.50	74.90	63.55	0.00	0.00
hbp3_30_4	104.15	100.70	100.90	104.15	100.70	100.90	2.55	11.65
hbp3_30_5	56.00	55.80	60.40	56.00	55.80	60.40	0.00	0.00
hbp4_15_1	68.05	82.35	70.15	68.05	82.35	70.15	0.00	0.00
hbp4_15_2	117.35	103.75	104.50	117.35	103.75	104.50	0.00	1.60
hbp4_15_3	70.35	74.40	95.00	70.35	74.40	95.00	0.00	0.40
hbp4_15_4	97.70	88.50	113.65	97.70	88.50	113.65	0.00	0.00
hbp4_15_5	142.60	147.95	151.65	142.60	147.95	151.65	0.00	0.55
hbp4_30_1	138.40	143.40	146.95	138.40	143.40	146.95	0.00	2.65
hbp4_30_2	194.50	166.85	206.75	194.50	166.85	206.75	0.00	1.15
hbp4_30_3	64.15	80.50	82.05	64.15	80.50	82.05	–	0.00
hbp4_30_4	128.50	154.35	123.45	128.50	154.35	123.45	0.00	0.00
hbp4_30_5	109.85	105.40	124.10	109.85	105.40	124.10	0.00	0.00
nBV	0	0	0	0	0	0	39	26

Table 11 – Makespans obtained by solution methods

Instance	Makespans							
	SCTSL	**SCTLL**	**SCTAL**	**LCTSL**	**LCTLL**	**LCTAL**	**M2**	**M2-SRH**
hbp1_15_1	1	1	1	1	1	1	1	1
hbp1_15_2	1	1	1	1	1	1	1	1
hbp1_15_3	1	1	1	1	1	1	1	1
hbp1_15_4	1	1	1	1	1	1	1	1
hbp1_15_5	1	1	1	1	1	1	1	1
hbp1_30_1	1	1	1	1	1	1	1	1
hbp1_30_2	1	1	1	1	1	1	1	1
hbp1_30_3	1	1	1	1	1	1	1	1
hbp1_30_4	1	1	1	1	1	1	1	1
hbp1_30_5	1	1	1	1	1	1	1	1
hbp2_15_1	3	3	3	3	3	3	3	3
hbp2_15_2	3	3	3	3	3	3	3	3
hbp2_15_3	4	4	3	4	4	3	3	3
hbp2_15_4	3	3	3	3	3	3	3	3
hbp2_15_5	3	3	3	3	3	3	2	3
hbp2_30_1	2	2	2	2	2	2	2	2
hbp2_30_2	2	2	2	2	2	2	2	2
hbp2_30_3	2	2	2	2	2	2	2	2
hbp2_30_4	2	2	2	2	2	2	2	2
hbp2_30_5	2	2	2	2	2	2	2	2
hbp3_15_1	6	6	6	5	5	5	6	5
hbp3_15_2	4	4	4	4	4	4	4	3
hbp3_15_3	6	6	6	5	5	5	5	5
hbp3_15_4	5	5	5	5	5	5	5	4
hbp3_15_5	5	5	5	4	4	4	4	4
hbp3_30_1	4	4	4	3	3	3	3	3
hbp3_30_2	4	4	4	3	3	3	3	3
hbp3_30_3	3	3	3	3	3	3	3	3
hbp3_30_4	4	4	4	3	3	3	3	3
hbp3_30_5	3	3	3	3	3	3	3	3
hbp4_15_1	7	7	7	6	6	6	7	7
hbp4_15_2	9	9	9	7	7	7	–	8
hbp4_15_3	7	7	7	7	7	6	–	6
hbp4_15_4	8	8	8	7	7	7	8	7
hbp4_15_5	8	8	8	8	8	8	–	7
hbp4_30_1	6	6	6	4	4	4	4	4
hbp4_30_2	6	6	6	5	5	5	6	5
hbp4_30_3	5	5	5	4	4	4	4	4
hbp4_30_4	6	6	6	4	4	4	5	4
hbp4_30_5	5	5	5	4	4	4	4	4
nBV	20	20	21	34	34	36	30	37

Table 12 – Total completion times obtained by solution methods

Instance	Total completion times							
	SCTSL	**SCTLL**	**SCTAL**	**LCTSL**	**LCTLL**	**LCTAL**	**M3**	**M3-SRH**
hbp1_15_1	7	7	7	7	7	7	7	7
hbp1_15_2	13	13	12	13	13	12	12	12
hbp1_15_3	11	11	11	11	11	11	10	10
hbp1_15_4	11	11	11	11	11	11	10	10
hbp1_15_5	15	15	14	15	15	14	14	14
hbp1_30_1	9	9	9	9	9	9	9	9
hbp1_30_2	8	8	8	8	8	8	8	8
hbp1_30_3	11	11	11	11	11	11	11	11
hbp1_30_4	11	11	11	11	11	11	10	10
hbp1_30_5	7	7	7	7	7	7	7	7
hbp2_15_1	42	42	42	42	42	42	40	40
hbp2_15_2	42	42	40	42	42	40	40	40
hbp2_15_3	45	45	43	45	45	43	42	42
hbp2_15_4	41	41	39	41	41	39	39	39
hbp2_15_5	32	32	32	32	32	32	30	32
hbp2_30_1	28	28	28	28	28	28	28	28
hbp2_30_2	39	39	39	39	39	39	38	38
hbp2_30_3	39	39	39	39	39	39	37	37
hbp2_30_4	30	30	30	30	30	30	30	30
hbp2_30_5	27	27	27	27	27	27	27	27
hbp3_15_1	67	67	67	67	67	67	67	65
hbp3_15_2	46	46	46	46	46	46	46	45
hbp3_15_3	64	64	66	64	64	66	64	64
hbp3_15_4	63	63	63	63	63	63	63	58
hbp3_15_5	54	54	53	54	54	53	53	53
hbp3_30_1	71	71	71	71	71	71	71	71
hbp3_30_2	67	67	67	67	67	67	67	64
hbp3_30_3	63	63	63	63	63	63	60	60
hbp3_30_4	64	64	64	64	64	64	62	62
hbp3_30_5	50	50	50	50	50	50	50	50
hbp4_15_1	84	84	84	84	84	84	104	84
hbp4_15_2	101	101	101	101	101	101	104	99
hbp4_15_3	91	91	90	91	91	90	91	87
hbp4_15_4	97	97	97	97	97	97	101	97
hbp4_15_5	106	106	106	106	106	106	106	103
hbp4_30_1	109	110	109	109	110	109	110	109
hbp4_30_2	140	140	140	140	140	140	138	131
hbp4_30_3	98	98	95	98	98	95	–	91
hbp4_30_4	109	109	107	109	109	107	–	109
hbp4_30_5	98	98	97	98	98	97	94	91
nBV	14	13	19	14	13	19	26	38

2.6 Case study

In this section, we present a case study with an instance that was generated based on real life data that for reasons of industrial secret could not be provided for tests. It is important to remark that this instance is not an actual client order, but all its data are very similar to the data that we could find in practice. The instance generated, termed Instance 1, is defined in Table 13.

Table 13 – Instance 1

Number of beam types			3		
Number of molds			15		
Number of periods			4		
Molds length (m)			60		
Type 1	Cure time		1		
	Number of Beams		4		
	Lengths (m)	2.9	3.2	4.6	7.15
	Demands	13	35	34	22
Type 2	Cure time		2		
	Number of Beams		4		
	Lengths (m)	2.9	3.95	6	6.9
	Demands	34	26	9	31
Type 3	Cure time		3		
	Number of Beams		4		
	Lengths (m)	3.95	5.7	5.95	6.9
	Demands	26	13	28	9

Instance 1 admits 1047 maximal patterns on forms of length 60m were generated plus pattern P_0, resulting in a total of 1048 patterns. Regarding only q_c-maximal pattern, there were 381 patterns generated plus pattern P_0, 382 totally. In the Gantt charts in this section, each component represents a pattern characterized by a label that represents its index and a color which states its type.

To evaluate the performance of the solutions obtained we used the following indicators:

- Total capacity: the sum of capacity of all molds available along the time horizon, measured in meters;
- Idle capacity: the sum of idle capacity of all molds along the time horizon, measured in meters;
- Productive capacity loss: the percentage of idle capacity along the time horizon;
- Concrete waste: concrete in the molds that where not used for beam production, measured in meters;

- Beam surplus: number of beams that were produced in addition to the quantity demanded.

Note that the total concrete waste multiplied by the number of periods in which such concrete were in the molds is equal to the total idle capacity.

2.6.1 Solving the case study with all maximal patterns

After solving Instance 1 by model (M1) we get the feasible solution in Figure 2, with objective function value of 1.05m total idle capacity, in 3,600 seconds with a gap of 14.19% and 0.35m of real waste of concrete. We can see in Table 14 that all molds were well used, 99.97% of their capacity was used along all time periods available. A large quantity of beam surplus, i.e. more beams than the ordered demand, were fabricated. Details can be seen in Table 15.

Figure 2 – Case Study: solution obtained with model (M1)

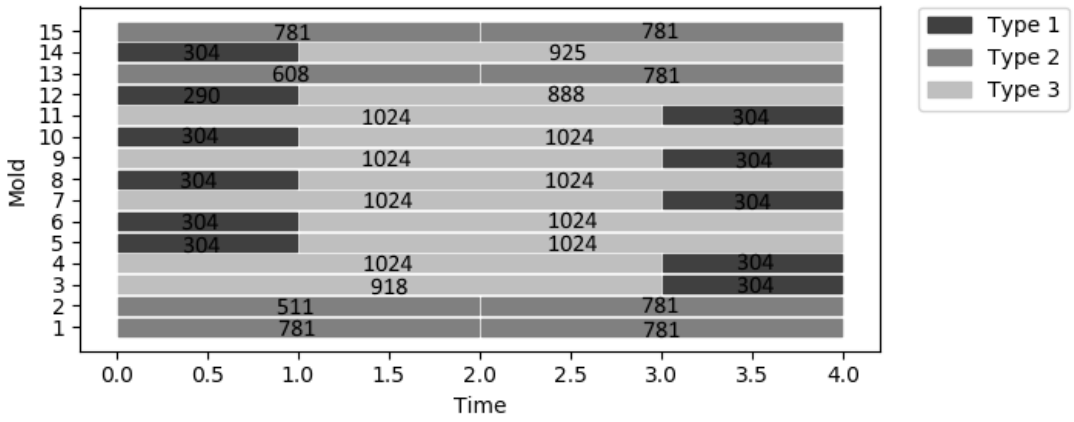


Table 14 – Case Study: mold usage of solution of model (M1) along the time horizon

Period	Total capacity	Idle capacity	Productive capacity loss
1	900	0.10	0.01%
2	900	0.35	0.04%
3	900	0.35	0.04%
4	900	0.25	0.03%
Total	3600	1.05	0.03%

Table 15 – Case Study: beam surplus from solution for instance by model (M1)

Type	Length 1	Length 2	Length 3	Length 4
1	48	5	9	0
2	1	2	0	0
3	2	1	1	34

When solving Instance 1 with model (M2) we get the optimal solution shown in Figure 3, with objective function value of 3 periods, in 5.7 seconds. The idle capacities in the molds on the periods 1, 2, 3 and 4, respectively, were 20.15m, 19.6m, 18.75m and 0m. As we can see in Table 16, the molds were not so efficiently used as in the solution of model (M1), but we got a reduction of 1 time period for the demand production. 97.83% of mold capacity was used along 3 time periods out of the 4 time periods available. We got 30.1m waste of concrete for this solution and 58.5m of idle capacity. Unlike model (M1), the solution of model (M2) produced a small quantity of beam surplus, as we can see in Table 17.

Figure 3 – Case Study: solution obtained with model (M2)

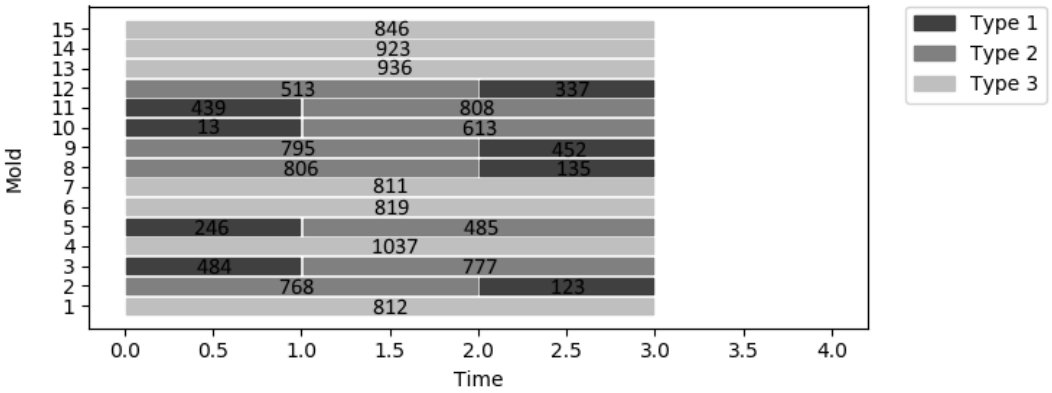


Table 16 – Case Study: mold usage of solution of model (M2) along the time horizon

Period	Total capacity	Idle capacity	Productive capacity loss
1	900	20.15	2.24%
2	900	19.6	2.18%
3	900	18.75	2.08%
4	–	–	–
Total	2700	58.5	2.17%

Table 17 – Case Study: beam surplus from solution obtained with model (M2)

Type	Length 1	Length 2	Length 3	Length 4
1	1	1	0	0
2	0	0	0	0
3	0	1	0	0

When solving Instance 1 with model (M3) we get the optimal solution shown in Figure 4, with objective function value equal to 45 periods, in 1990.8 seconds. As we can see in Table 18, the molds in the solution of model (M3) were almost so efficiently used as in the

solution of model (M1). 99.86% of mold capacity was used along the 4 time periods available. We got 1.9m of waste of concrete for this solution and 5.2m of idle capacity. Like model (M2), model (M3) solution produced a small quantity of beam surplus, 7 beams more were produced in comparison to model (M2), as we can see in Table 19.

Figure 4 – Case Study: solution obtained with model (M3)

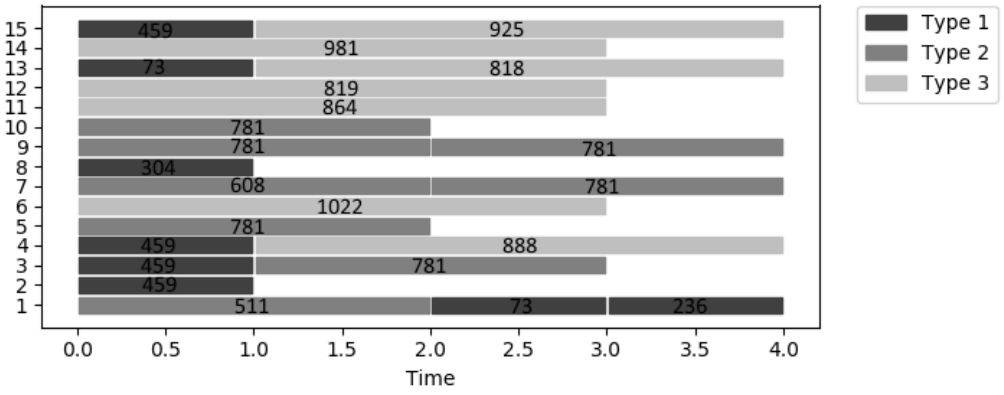


Table 18 – Case Study: mold usage of solution of model (M3) along the time horizon

Period	Total capacity	Idle capacity	Productive capacity loss
1	900	1.05	0.12%
2	900	1.65	0.18%
3	900	1.65	0.18%
4	900	0.85	0.09%
Total	3600	5.2	0.14%

Table 19 – Case Study: beam surplus from solution obtained with model (M3)

Type	Length 1	Length 2	Length 3	Length 4
1	0	2	1	0
2	1	2	0	0
3	0	0	1	1

2.6.2 Solving the case study with size-reduction heuristic

Regarding the solution for the case study obtained with using model (M1) with the SRH, which we can see in Figure 5, we got an objective function value of 1.35m total idle capacity, after 3,600 seconds of running time with final gap of 18.52%. It involved 0.45m of waste of concrete. In Table 20, the molds were efficiently used, 99.96% of their capacity was

used along all time periods available. A large amount of beam surplus was produced, see Table 21.

Figure 5 – Case Study: solution obtained with model (M1) with SRH

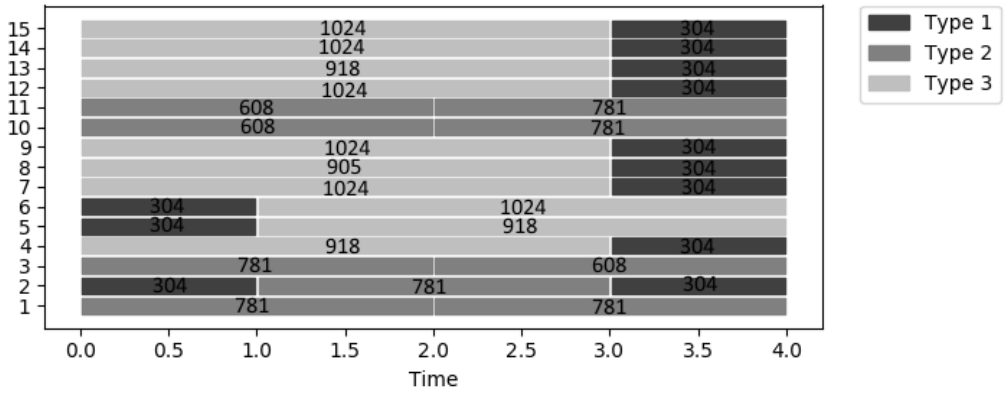


Table 20 – Case Study: mold usage of solution of model (M1) with SRH along the time horizon

Period	Total capacity	Idle capacity	Productive capacity loss
1	900	0.35	0.04%
2	900	0.45	0.05%
3	900	0.45	0.05%
4	900	0.10	0.01%
Total	3600	1.35	0.04%

Table 21 – Case Study: beam surplus from solution obtained with model (M1) with SRH

Type	Length 1	Length 2	Length 3	Length 4
1	47	13	14	2
2	8	4	3	2
3	1	1	2	25

In Figure 6 we can see the solution returned by model (M2) using the SRH for the case study. The solution has a makespan value of 3 periods, having required 2.32 seconds of running time and showing a final gap of 0%. It involved 27.85m of waste of concrete and 53.85m of idle capacity along the time horizon of 3 periods. In Table 22, the molds were not as efficiently used as model (M1) using the SRH, 98.01% of their capacity was used along all time periods available. A small quantity of beam surplus was produced, see Table 23, only 1 unit of beam of length 4 from type 1, and 1 unit of beam of length 4 from type 3.

Figure 6 – Case Study: solution obtained with model (M2) with SRH

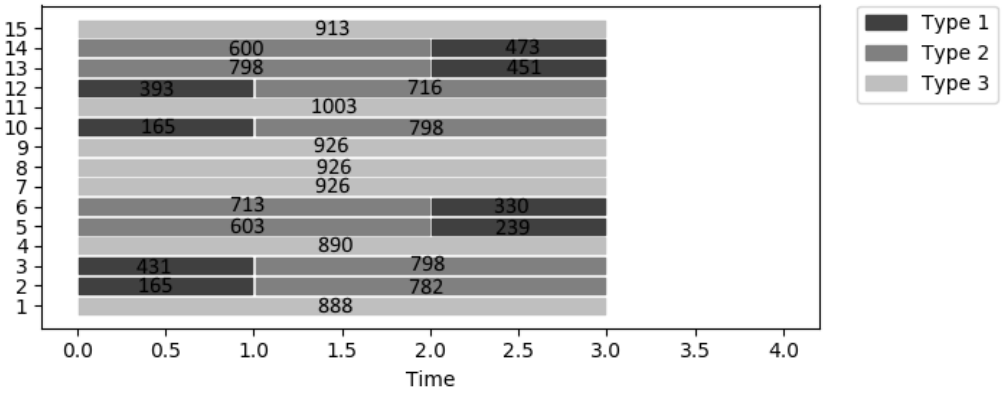


Table 22 – Case Study: mold usage of solution of model (M2) with SRH along the time horizon

Period	Total capacity	Idle capacity	Productive capacity loss
1	900	17.55	1.95%
2	900	18.40	2.04%
3	900	17.90	1.99%
4	–	–	–
Total	2700	53.85	1.99%

Table 23 – Case Study: beam surplus from solution obtained with model (M2) with SRH

Type	Length 1	Length 2	Length 3	Length 4
1	0	0	0	1
2	0	0	0	0
3	0	0	0	1

In Figure 7, we show the solution returned by model (M3) using the SRH for the case study. The solution has a total completion time value of 45 periods, after 18.01 seconds of running time with final gap of 0%. Note that mold 12 was not used in such solution. It involved 7.20m of waste of concrete and 13.90m idle capacity along the 4 periods time horizon. In Table 24, the molds were almost as efficiently used as model (M1) using the SRH, 99.61% of their capacity was used along all 4 time periods available. A small quantity of beam surplus was produced, detailed in Table 23.

Figure 7 – Case Study: solution obtained with model (M3) with SRH

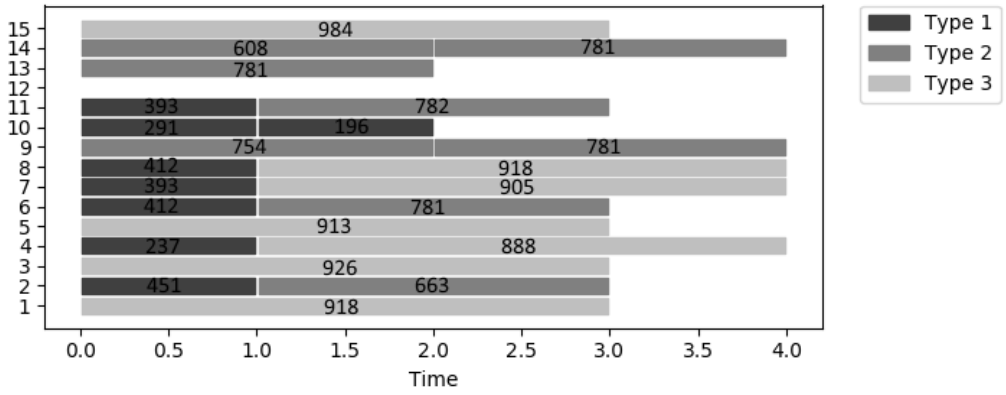


Table 24 – Case Study: mold usage of solution of model (M3) with SRH along the time horizon

Period	Total capacity	Idle capacity	Productive capacity loss
1	900	3.95	0.44%
2	900	5.70	0.63%
3	900	3.80	0.42%
4	900	0.45	0.05%
Total	3600	13.90	0.39%

Table 25 – Case Study: beam surplus from solution obtained with model (M3) with SRH

Type	Length 1	Length 2	Length 3	Length 4
1	1	1	0	1
2	1	0	0	1
3	0	1	1	0

2.6.3 Comparing solutions obtained with models and priority rules

As we can see in Table 26, there was a significant reduction on execution time when we solved the models using the SRH. We also observed empirically a large reduction in memory usage: since the number of patterns it less numerous, the number of variables and constraints are drastically reduced making the model much smaller than the one with all maximal patterns. Regarding solution quality of model using SRH, there was not a significant difference as compared to the complete model. The makespans and total completion times were all the same for each model using SRH or not, changes were only detected on idle capacities and concrete waste.

Regarding the priority rules, we can see that their solutions were very much alike among each other. Although they spend an insignificant execution time, the quality of solution

in terms of idle capacities and concrete waste are too low when compared to those of models (M1) and (M3). We can see that the quality of solutions obtained regarding makespan and total completion time are not so low when compared to those of the mathematical models. Nevertheless, none of the priority rules could find neither the optimal makespan nor the best total completion time found by the models for Instance 1.

Table 26 – Comparisons among solutions for the Case Study

	Idle capacity	**Concrete waste**	**Makespan**	**Total completion time**	**Execution time (s)**
M1	**1.05**	**0.35**	4	60	3600.00
M1-SRH	1.35	0.45	4	60	3600.00
M2	58.50	30.10	**3**	**45**	5.70
M2-SRH	53.85	27.85	**3**	**45**	2.32
M3	5.20	1.90	4	**45**	1990.80
M3-SRH	13.90	7.20	4	**45**	18.01
SCTSL	57.10	29.60	4	50	**0.00**
SCTLL	64.00	31.55	5	50	**0.00**
SCTAL	60.20	32.30	5	47	**0.00**
LCTSL	57.10	29.60	4	50	**0.00**
LCTLL	64.00	31.55	4	50	**0.00**
LCTAL	60.20	32.30	4	47	**0.00**

2.6.4 Symmetry breaking constraints

As we can see from the solutions of all proposed models, there may be many symmetric solutions. For example, if we change the order of production of the patterns in mold 13 of the solution shown in Figure 2 we would get the same solution with a different representation. We call this a *period-based symmetry*. The same occurs when we move the patterns produced in a mold to another one, for example, if we move the produced patterns on mold 14 to mold 15 and vice versa of the solution shown in Figure 2 we would get the same solution with a different representation. We call this a *mold-based symmetry*. In order to circumvent this problem and, consequently, improve the running times of the models we define two sets of constraints for breaking both period and mold-based symmetries:

$$\sum_{i \in Q^*(m)} i\, x_i^{m,t} \leq \sum_{i \in Q^(m)} i\, x_i^{m,t+1} + r x_0^{m,t+1}, \qquad m = 1,\ldots,M,\ t = 1,\ldots,T-1 \tag{2.21}$$

$$\sum_{t=1}^{T} \sum_{i \in Q^*(m)} i\, t\, x_i^{m,t} \leq \sum_{t=1}^{T} \sum_{i \in Q^*(m+1)} i\, t\, x_i^{m+1,t}, \qquad m = 1,\ldots,M-1. \tag{2.22}$$

Constraints (2.21) state that patterns will be fabricated in a crescent order of indexes in the molds. Constraints (2.22) define that molds will be used for patterns production in a crescent order defined by the value of $\sum_{t=1}^{T} \sum_{i \in Q^*(m)} i\, t\, x_i^{m,t}$ for each mold m. Thus, it is easy to see that Proposition 2.6.1 is true.

Proposition 2.6.1. *Addition of symmetry breaking constraints does not modify the optimal solution value of any of the proposed models.*

Although considering symmetric solutions increases drastically the problem's search space and symmetry breaking constraints reduce the number of solutions greatly, we noted, during preliminary computational tests, that the usage of these constraints made the models much bigger and hard to solve. The models required greatly more memory and execution time than before and their insufficient solution performance led us to not consider such constraints in our larger computational tests.

2.7 Conclusions

In this chapter, we introduced a novel variant of cutting sequencing problems, called HPPBMPP, along with four integer linear programming models for its solution, one for idle capacity minimization, two for makespan minimization, and one the total completion time minimization.

We proposed a size reduction heuristic consisting in the reduction of the number of patterns, thus cutting down, drastically, the number of constraints and variables, which led us to lighter and easier to solve problems. We also created 6 priority rules based on classic scheduling constructive heuristics as a way of finding feasible solutions really fast. With big-size instances they showed to be a good method compared even to the mathematical models, which sometimes spend too much time and/or memory to get to feasible solutions. Such constructive heuristics can be a promising way to find initial solutions for some metaheuristic method.

We proposed a set of randomly generated benchmark instances for the HPPBMPP, all of which are based on data arising from a real-life application. Computational tests were performed using off-the-shelf optimization software, in order to assess the effectiveness of the mathematical models. The results suggest that model (M1) was relativity easier to solve compared to the other models, since as the diversity of beam lengths increases, so does the likelihood of the model finding a solution with zero idle capacity. Thus, in practice, simply

filling the molds to the maximum capacity is not so attractive since a lot of beams produced will be stocked and the production line will not be so flexible with respect to possible changes in the plan.

As regards the models that minimize makespan and total completion time, in general, they were able to solve the great majority of instances with up to two types of beams to optimality within short computing times. When solving instances with 3 or 4 types of beams, the models tended to require excessive memory and/or running time. Tests carried out using the size reduction heuristic with a specific type of patterns showed to be an interesting way of exploring the proposed models presenting good solutions with less memory and running time requirements.

In the case study we were able to find good solutions for the instance explored with mathematical models using SRH or not. Model (M1) tends to find solutions with extremely low idle capacity and concrete waste, thus with a high makespan an total completion time. Model (M2) found solutions with an optimal makespan and optimal total completion time, and higher idle capacities than model (M1), around 2% of total capacity loss, which is still low compared to industry loss that is around 5%-10%. Model (M3) found the optimal total completion and provided good solutions in terms of molds usage. The production line tended to be nevertheless excessively unbalanced aggravating the makespan.

Natural directions for future work, in the context of exact models, entail the use of more sophisticated solution approaches (such as column generation), or different modeling strategies. Alternatively, the development of heuristic algorithms for producing high-quality solutions can be useful when handling large instances. The practical nature of the problem also suggests the study of variants of the HPPBMPP, possibly including scheduling constraints, the reuse of leftover material, delivery dates, or dynamic demand.

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

Abstract

In this work, we introduce a new 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 that were carried out for the proposed solution methods 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 proposed genetic algorithm was able to find good-quality solutions for large-sized instances within short computing times.

Keywords: precast beams; modular construction; integer linear programming; metaheuristics; genetic algorithms.

3.1 Introduction

Nowadays, concrete precast production is increasingly trending in constructions sites. There are great advantages of using such kind of production, such as better and cheaper elements, and a 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.

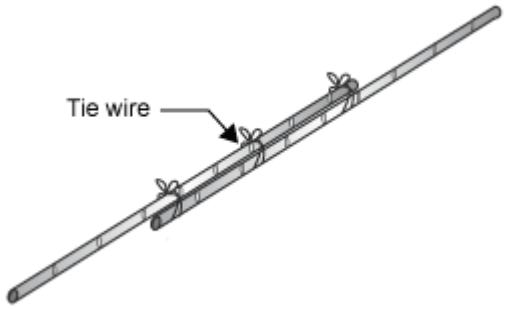
These beams are heterogeneous in the sense that they can vary with respect to curing time, length and the number of traction elements used. We refer to the problem of planning the production of such beams to fulfill the clients demand within a given time horizon as the Heterogeneous Precast Beams Multiperiod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

Araujo *et al.* (2019) 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 for production makespan minimization and one model for total completion time minimization. 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, in the problem. We divide the bars in two groups: standard bars and *leftovers*. Standard bars are new bars of standardized lengths, and leftovers are a type of bar that cannot be 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, illustrated in Figure 8, consists in merging two or more leftovers to create a larger bar that can be cut to produce a bar of appropriate length that will be used in beam production. In this work, we only consider overlapping only two bars. To the best of our knowledge, the consideration of overlapping bars has not been previously studied in the revised literature.

Figure 8 – Illustrative example of leftover overlapping process



Source: Modified from <https://emedia.rmit.edu.au/dlsweb/Toolbox/buildright/content/bcgbc4010a/10_floor_systems/03_concrete_slab_reinforcement/page_006.htm>

We call the problem of integrating the cutting process of bars or overlapping bars that will be "packed" in the molds for the production of a client order of beams in a single production planning 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 Arenales *et al.* (2015), which deals with the cutting stock/leftover problem, and on the model by Araujo *et al.*

(2019) 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 have a stock that is not replenished over all entire time horizon.

Studying this problem is interesting in practice since 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, and minimizing bars loss. A better usage of bar stock may results in a reduction exceeding bars in the construction site, which can improve the production flow. Furthermore, the reduction of concrete and bar loss may lead to a positive impact in the environment. An optimized process allows factories to accept additional orders due to shorter lead times. Also, 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 (ARAUJO *et al.*, 2019) 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. Therefore, future computational tests results may show that the ICP-HPBMPP may 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 chapter is organized as follows. In Section 3.2 we discuss the literature of similar problems to the ICP-HPBMPP. In Section 3.3 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 3.4 we present three constraint programming models for the generation of packing, cutting and overlapping patterns. In Section 3.5 we propose a genetic algorithm for the problem under study. In Section 3.6 we discuss several computational experiments conducted based on instances generated artificially and discuss the results of the proposed solution methods. In Section 3.7 we discuss the conclusions and contributions of this chapter, as well as point out research gaps and suggest future work.

3.2 Literature review

To the best of our knowledge there is no mathematical model in the literature for the ICP-HPBMPP, although it holds considerable 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.

To give some examples concerning the one-dimensional cutting and packing problems (C&P), the studies of Gilmore and Gomory (1961) and Gilmore and Gomory (1963) 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. Stadtler (1990) studied the 1DCSP proposing a heuristic 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 case of study of a manufacturer of aluminum profiles. Dyckhoff (1990) introduced a typology of C&P problems unifying notions in the literature to guide further research on particular types of those problems. Vance (1998) proposed two different branch-and-price approaches to find optimal solutions to the 1DCSP. Wäscher *et al.* (2007) presented a new typology to categorize the types of C&P problems in the literature between the years 1995 and 2004, introducing new categorization criteria. Trkman and Gradisar (2007) proposed a model for the multiperiod one-dimensional cutting stock problems (M1DCSP) considering the use of objects/leftovers in stock. Poldi and Arenales (2010) proposed an integer linear model for the M1DCSP, implemented a column generation to solve the linear relaxation, and developed two rounding heuristics for finding integer solutions to the problem. Melega *et al.* (2018) proposed a mathematical model for the general integrated lot-sizing and cutting stock problem, and performed a vast classification of the literature of that problem, providing directions for future research.

Regarding the C&P problems and optimization approaches in precast production, de Castilho *et al.* (2007) described the problem of minimizing production costs for slabs of precast prestressed concrete joists and introduced a genetic algorithm to solve it. Prata *et al.* (2015) proposed an integer linear programming model for multiperiod production planning of precast concrete beams, which can be seen as a special case of the HPBMPP. Arenales *et al.* (2015) introduced a mathematical model for the cutting stock/leftover problem and suggested a column generation technique for finding the problem's linear relaxation solution. Vassoler *et al.* (2016) proposed a mathematical model based on 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. Araujo *et al.* (2019) proposed several integer linear programming models for the Heterogeneous Prestressed Precast Beams Multiperiod Production Planning Problem, argued its NP-hardness

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 approached. Wang *et al.* (2018) 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 considering uncertainty in processing time for each operation. The authors validated the model with a case study.

The problem which we study in this work is the integration of the cutting stock/left-over problem proposed by Arenales *et al.* (2015) and the HPBMPP introduced by Araujo *et al.* (2019).

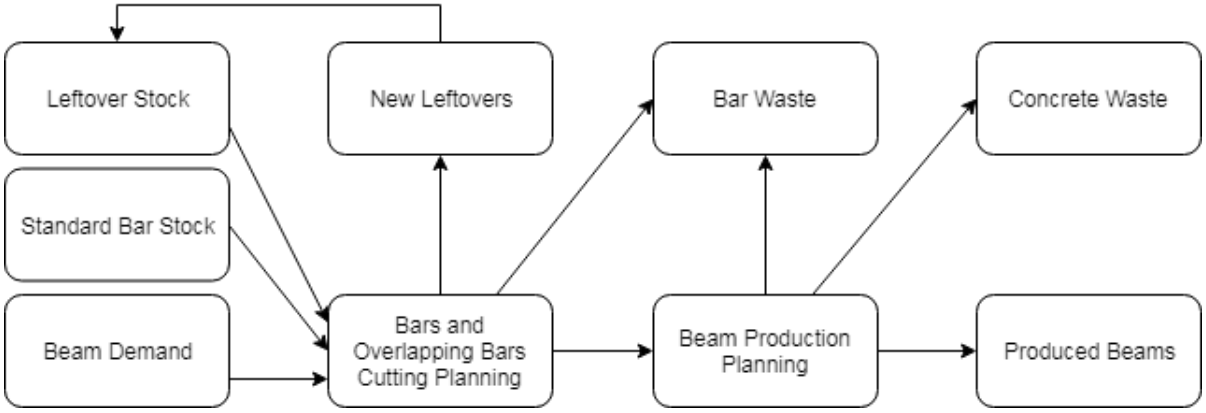
3.3 Problem statement

In this section we formally define the ICP-HPBMPP and propose an integer linear programming model for its solution based on the models proposed by Arenales *et al.* (2015) for the Cutting Stock/Leftover Problem (CSLP) and Araujo *et al.* (2019) 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*, which is understood as a piece of bar that can be cut or overlapped in the 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 production cannot be greater than a given time horizon. After the process of cutting of the bars, they are packed in the molds in order to produce the beams, under the condition 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 9.

Figure 9 – Cutting and packing production flowchart



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 cut will be packed in the molds for the beam production along the given time horizon. After the production of all beams demanded there will usually be concrete waste of the beams and additional loss of bars.

3.3.1 Integer linear programming model

In order to define a model for the ICP-HPBMPP, we make use of the same parameters defined in (ARAUJO *et al.*, 2019), 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 $m = 1,\ldots,M$;
- $P_i = (c_i,(a^i_1,\ldots,a^i_{q_c}))$: packing pattern, $i = 1,\ldots,r$, $c = 1,\ldots,C$. 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 $P_i = \left(\bar{c}, (\bar{a}_1, \ldots, \bar{a}_{q_c})\right)$, 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 $c = \bar{c}$, then $\mathcal{N}_i(c,k) = \bar{a}_k$, with $k \in \{1, \ldots, q_c\}$; otherwise, $\mathcal{N}_i(c,k) = 0$, for any k.
- $u(P_i)$: capacity used by P_i, i.e. $u(P_i) = \sum_{k=1}^{q_{\bar{c}}} l(\bar{c},k) \cdot P_i(\bar{c},k)$, with $i = 1, \ldots, r$.
- c_i: beam type corresponding to the pattern P_i, $c_i \in \{1, \ldots, C\}$ and $i = 1, \ldots, r$.
- 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 $\bar{c}$, given by $t_{\bar{c}}$.

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

- $Q(m)$: set containing the indices of the patterns in $\mathscr{P}$ 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 $j \in \{1, ..., R\}$, 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;

- $\mathscr{L}_1, \ldots, \mathscr{L}_\Gamma$: mold lengths. Note that this data narrows the types of cutting, and overlapping patterns;
- $G(\mathscr{L}_\gamma)$ = set of molds which are of length $\mathscr{L}_\gamma$, $\gamma = 1, \ldots \Gamma$;
- 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 $\mathscr{L}_\gamma$.
- $I_h = (w_h, (a_1^h, \ldots, a_\Gamma^h, a_{\Gamma+1}^h, \ldots, a_{\Gamma+V}^h))$: 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 $\mathscr{L}_1, \ldots, \mathscr{L}_\Gamma$ 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 $\mathscr{L}_{\gamma_\mu}$, 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 $\mathscr{L}_\gamma$ 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 $\mathscr{L}_\gamma$ 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:

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

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

$y_{h,w}$: 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 integer decision variables. We present the integer linear programming model proposed for the ICP-HPBMPP as follows:

(ICP) min

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

s.t.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$v = 1, \ldots, V,$$
$$h \in H_w(v). \tag{3.15}$$

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

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 to 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. In terms of multi-objective optimization, i.e., when we consider multiple objective functions to be optimized, each term of (3.1) can be seen 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 function (3.1) is, therefore, a Pareto optimum (MARLER; ARORA, 2004). Minimizing (3.1) means minimizing the total length of bars that are cut and are neither being used for bars production nor for leftover generation, plus the makespan, considering their respective weights λ_i, with $i = 1, \ldots, 4$.

Constraints (3.2) ensure that at most one pattern shall 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 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 bar leftovers cut plus the number of leftover bars used to produced bars via overlapping do 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 do not exceed the stock. Constraints (3.11) force that the amount of bars required to produced 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 $q = \sum_{i=1}^{C} q_c$. 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, that are typically the more numerous in practice, by using only maximal packing patterns, as introduced by Araujo *et al.* (2019). 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 3.3.1. *Restricting the model (ICP) to using only maximal packing patterns does not modify its set of optimal solutions.*

Proof. Given an optimal solution to model (ICP) that is composed by non-maximal packing patterns we claim that replacing the non-maximal packing patterns with maximal ones that contain such patterns will not have an impact on the makespan. Indeed, the actual number of periods used to fulfill the demand will remain unaffected, given that all packing patterns of a given type have the same associated curing time. In the same way, there will be no changes to the cutting and overlapping patterns used in the optimal solution since the number of bars needed for the beam production will remain unchanged. □

3.3.2 NP-hardness

To argue the ICP-HPBMPP complexity 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 the actual problem into a HPPMBPP problem for makespan minimization up to a constant multiplicative factor. Consequently, the ICP-HPBMPP can be seen as a generalization of HPPMBPP, which is already known to be NP-Hard (ARAUJO *et al.*, 2019).

3.3.3 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 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:

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

A lower bound for the optimal value of model (ICP) is given by Equation (3.17).

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

The first part of Equation (3.17) corresponds to a the lower bound for the makespan, while the second part stands for the minimum waste of bar to produce the beam demand when using molds of some fixed length $\mathscr{L}_\gamma$.

3.4 Patterns generation

Instead of carrying out exhaustive enumerations, we generated the desirable packing and cutting patterns for a given instance using constraint programming models.

3.4.1 Packing patterns generation

Consider the following notation, in addition to the notation presented in Section 3.3:

- 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 $K = 6$.
- $v_i \in \{1,\ldots,C\}$ is a decision variable that corresponds to the type of beam used by the pattern P_i.

- $\gamma_i \in \{1,\ldots,\Gamma\}$ is an 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 $[\mathscr{N}_i(v,j)]_{j=1}^{q_v}$.
- $P_i = \left(v_i, (A^i_1,\ldots,A^i_{q_v})\right)$: the generated pattern.

For the generation of a packing pattern P_i we present the model, adapted from (ARAUJO *et al.*, 2019), as follows:

$$1 \leq v_i \leq C, \tag{3.18}$$

$$1 \leq \gamma_i \leq \Gamma, \tag{3.19}$$

$$A^i_j = 0, \qquad \text{if } v_i = c, \qquad c = 1,\ldots,C, \quad j = q_c+1,\ldots,K \tag{3.20}$$

$$\mathscr{L}_m - \min_{j=1,\ldots,q_c}(l(c,j)) < \sum_{j=1}^{q_c} l(c,j)\cdot A^i_j \leq \mathscr{L}_m, \quad \text{if } (v_i = c \wedge \gamma_i = m), \quad c = 1,\ldots,C, \quad m = 1,\ldots,\Gamma, \tag{3.21}$$

$$A^i_k \in \mathbb{Z}_+, \qquad k = 1,\ldots,K. \tag{3.22}$$

Constraint (3.18) implies that the pattern type has domain $\in \{1,\ldots,C\}$. Constraint (3.19) defines the length of the molds in which the generated pattern should be maximal. Constraint set (3.20) 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 (3.21) 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 (3.18) - (3.22).

3.4.2 Cutting patterns generation

As in the case of packing patterns generation, we adopted constraint programming in on order to generate the cutting patterns which will be used as a part of the integer model input.

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 $\mathscr{L}_i$ 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 $i \in \{\Gamma+1,\ldots,\Gamma+V\}$;
- $I_h = \big(w_h, (A_1^h,\ldots,A_\Gamma^h,A_{\Gamma+1}^h,\ldots,A_{\Gamma+V}^h)\big)$: the generated pattern.

We define the proposed constraint model for generating a cutting pattern H_h below:

$$1 \leq w_h \leq W+V, \tag{3.23}$$

$$\sum_{i=1}^{\Gamma} \mathscr{L}_i \cdot A_i^h + \sum_{i=1}^{V} b_{W+i} \cdot A_{\Gamma+i}^h \leq \text{element}(w_h, b), \tag{3.24}$$

$$\#\{i \in \{\Gamma+1,\ldots,\Gamma+V\} | A_i^h > 0\} = 1, \tag{3.25}$$

$$A_i^h = 0, \qquad \text{if } w_h > W, \qquad i = \Gamma+1,\ldots,\Gamma+V \tag{3.26}$$

$$A_i^h \in \mathbb{Z}_+, \qquad i = 1,\ldots,\Gamma+V. \tag{3.27}$$

Constraint (3.23) defines the integer decision variable w domain. Such variable 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 (3.24) states that the length of items cut in the pattern is 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 (BELDICEANU; CARLSSON, 2018). Constraint set (3.25) implies that a cutting pattern only generates one type of leftover. Constraint (3.26) imply that a leftover does not generate more leftovers. We utilized the CPLEX CP Optimizer to enumerate all the solutions of model (3.23) - (3.27).

3.4.3 Overlapping patterns

In order to allow the possibility of overlapping bars to the problem, we recall that an overlapping pattern $\mathbb{O}_\mu$ as the tuple:

$$\mathbb{O}_\mu = (\gamma_\mu, (a_1^\mu,\ldots,a_V^\mu)) \tag{3.28}$$

Note that γ is associated to the length of 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 to 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 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 $\mathscr{L}_{\gamma}$.
- $\mathbb{O}_{\mu} = \left(\gamma, (A_1^{\mu},\ldots,A_V^{\mu})\right)$: the generated pattern.

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

$$1 \leq \gamma_{\mu} \leq \Gamma, \tag{3.29}$$

$$\sum_{i=1}^{V} A_i^{\mu} b_{W+i} \geq \mathscr{L}_{\gamma_{\mu}} + \varepsilon, \tag{3.30}$$

$$\sum_{i=1}^{V} A_i^{\mu} = 2, \tag{3.31}$$

$$f = \mathscr{L}_{\gamma_{\mu}} - \sum_{i=1}^{V} A_i^{\mu} b_{W+i}. \tag{3.32}$$

Constraint (3.29) ensures that the length of the bar produced is one of the possible mold length. Constraint (3.30) forces that the total length of the chosen leftovers are greater than the length of the bar produced via overlapping plus an ε that means the length of the bars used in the overlapping. Constraint (3.31) defines that only 2 leftovers are used in the production of the bar made via overlapping. Constraint (3.32) defines the bar waste produced in the overlapping pattern generated.

The constraint programming model for overlapping pattern generation is sufficiently flexible to accommodate the production planner's necessities. In a more general way, we could suppose that a bar made via overlapping can 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.

3.5 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.

3.5.1 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 10, we show a generic scheme of the solution representation, in which the number of columns is exactly $r+H+O$.

Figure 10 – Solution representation

a_1	a_2	...	$a_{r+H+O-1}$	a_{r+H+O}
x_1	x_2	...	$x_{r+H+O-1}$	x_{r+H+O}

In order to illustrate the solution representation we first present instance cwp000, generated randomly, in Table 27. Its respective packing, cutting, and overlapping patterns are presented in Tables 28, 29, and 30, respectively.

Table 27 – Instance cwp000 description

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

Table 28 – 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

Table 29 – 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 30 – 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

Note that ID is associated with the pattern indices. An optimal solution for the cwp000 instance is shown as the chromosome in Figure 11.

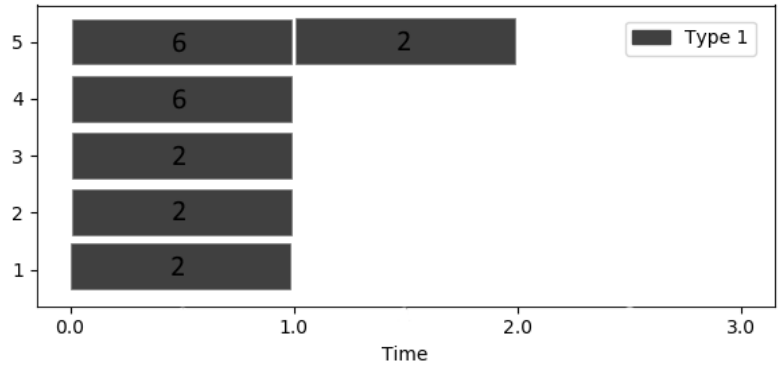
Figure 11 – Example of a feasible solution of instance cwp000

6	16	2	11	15
2	2	4	2	1

For the solution in Figure 11 we obtain an objective function value of 2.1, with makespan of 2 periods and bar waste of 0.1m. Figure 12 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 distinct length of mold, we infer that packing pattern 2 is associated with molds of length 5.95m, and packing pattern 6 is associated with molds of length 11.95m. Therefore, we need to produce a total number of 2 bars of length 5.95m and 6 bars of length 11.95, since the beam type produced by each solution packing patterns 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.

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 represented encoded in the chromosome in Figure 11 is feasible.

Figure 12 – Gantt chart for an optimal solution of instance cwp000



3.5.2 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. 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.

3.5.3 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.

3.5.4 Crossover operators

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

Algorithm 1: Generate pseudo-random solution

input: Instance, Set of Packing Patterns, Set of Cutting Patterns, Set of Overlapping Patterns
output: 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* $\mathscr{L}_\gamma$ *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 $\mathscr{L}_\gamma$ that has yet not been selected.
        bars_needed $\leftarrow$ number of bars of length $\mathscr{L}_\gamma$ 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* $\mathscr{L}_\gamma$ *needed was not reached* **do**
        $ove_p \leftarrow$ random overlapping pattern that generates a bar of length $\mathscr{L}_\gamma$ that has not yet been selected.
        bars_needed $\leftarrow$ number of bars of length $\mathscr{L}_\gamma$ 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*

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 2, 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 2, is applied to it and all patterns with final frequencies equal to zero have their respective genes removed from the chromosome.

3.5.5 Mutation operator

The mutation of an individual consists of choosing one pattern p_1 that is in the solution, and in the addition of 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 the fixing phase to it. This process is frequently required in practice and is described in this next subsection.

3.5.6 Infeasible solution fixing

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

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 exceed 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 2. Each infeasibility type is treated in a particular procedure: Algorithms 5, 6, and 7, in Appendix A, are used to fix infeasibility type 1, 2, and 3, respectively.

Algorithm 2: Solution fixing procedure

input: Infeasible chromosome
output: Possible feasible chromosome
if *Infeasibility type 1 = true* **then**
  Call Algorithm 5;
else
  Call Algorithm 4;
end
if *Infeasibility type 2 = true* **then**
  Call Algorithm 6;
end
if *Infeasibility type 3 = true* **then**
  Call Algorithm 7;
end
return *chromosome*

The unnecessary packing patterns procedure, shown in Algorithm 4, in Appendix A, works like a solution treatment phase, which is not a necessary part of the solution fixing process, although applying such procedure we 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 13, 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 for offspring 2 and not for offspring 1. In Figure 14, we show an example of the 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.

Figure 13 – Crossover operators

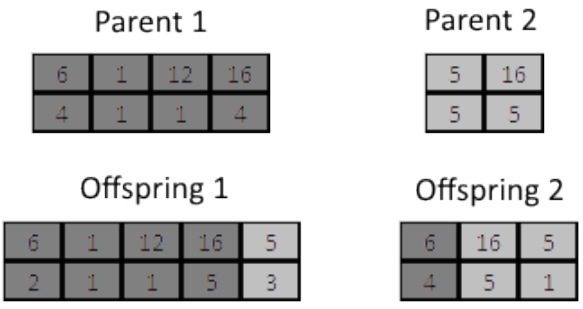
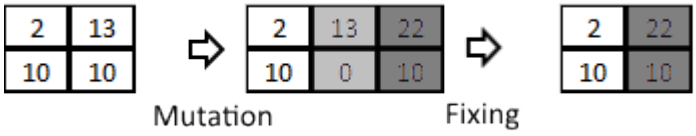


Figure 14 – Mutation operator and solution correction



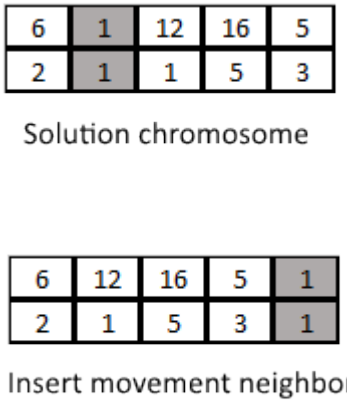
3.5.7 Population restart

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.

3.5.8 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 $k+1$ are moved one position to the right after the insertion of the k-th gene. In Figure 15 an insert movement neighbor is shown for a given solution after inserting 2nd gene in front of 5th gene.

Figure 15 – Insert movement



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 3.

Algorithm 3: Insert neighborhood

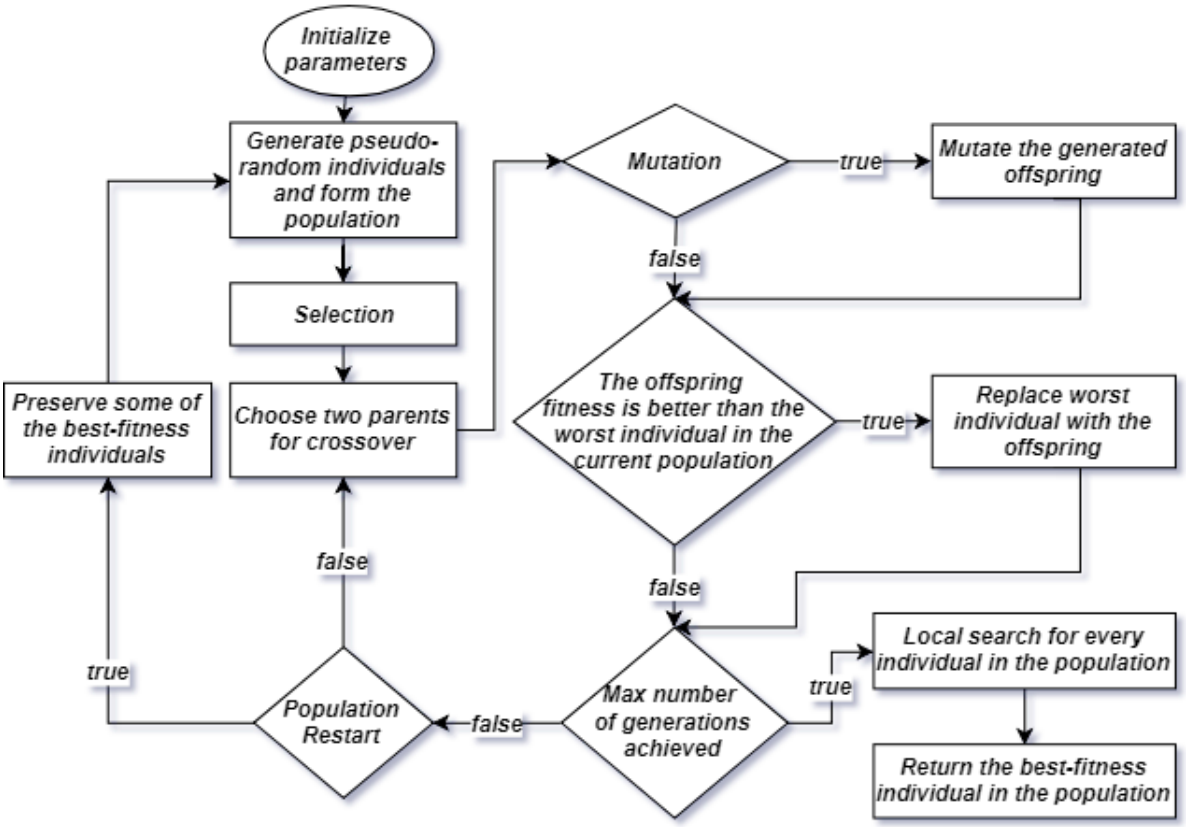
input: *InitialSolution*
output: *BestSolution*
BestSolution $\leftarrow$ *InitialSolution*;
for $i = 1, \ldots, n\ell - 1$ **do**
  for $k = k+1, \ldots, n\ell$ **do**
    neighbor $\leftarrow$ INSERT$(InitialSolution, i, k)$;
    if $makespan(neighbor) < makespan(BestSolution)$ **then**
      BestSolution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

3.5.9 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 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).

The proposed genetic algorithm can be seen as a steady-state model since only one new individual is generated per generation, even though we generate 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 16.

Figure 16 – Simplified flowchart of proposed genetic algorithm



3.6 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, to evaluate the solution methods proposed in this study.

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 work on a Linux Ubuntu 18.04 64bits machine with 8GB of memory and Intel Core i5-3470 CPU 3.20 GHz $\times 4$ 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 $i = 1, \ldots, 4$.

3.6.1 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 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 31 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.

Table 31 – Description of test instances

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

We consider mold capacities of 5.95m and 11.95m, while we take 1.12m, 1.45m, 2.35m, 2.5m, 2.65m, 2.95m, and 3.3m 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 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 by Equation 3.33 as follows:

$$T = \left\lceil 1.5 \cdot \frac{\sum_{i=1}^{C} t_c \cdot \left(\sum_{k=1}^{q_c} l(c,k) \cdot d(c,k) \right)}{\sum_{m=1}^{M} L_m} \right\rceil. \tag{3.33}$$

For all instances, we consider an unique length of new bars as 12m and the possible lengths of bar leftovers as 2m, 5m, 6m, and 8m. We do not vary such lengths along the test instances since, in practice, it is expected that they are standardized. To generate realistic bar stocks we introduce an upper bound for the number of bars needed to fulfill the beam demand as UB, defined in Equation (3.34):

$$UB = 2 \cdot T \cdot M \cdot \max_{D_c=1,\ldots,C} \{D_c\}. \tag{3.34}$$

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

3.6.2 Computational experiments with the mathematical model

In this subsection we carry out computational tests with the benchmark instance set that we generated following the scheme described in Subsection 3.6.1. In Table 32 we show the results of the computational experiments for the model (ICP) and its linear relaxation. The solution time was limited to 3,600 seconds. As regards to notation in Table 32, 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.

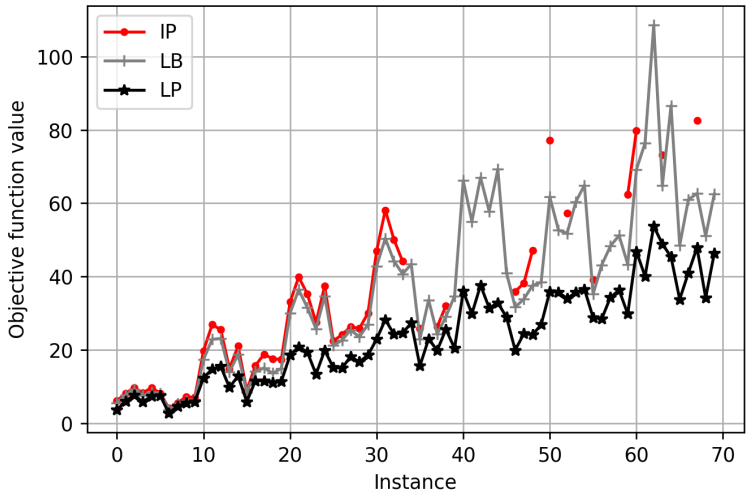
We can see in Table 32 that the linear relaxation of all instances could be solved, with the average time of 53.21 seconds, and with 624.61 seconds being the longest time to get to the optimal solution. On the other hand, only 11 instances could be solved to optimality by the integer programming model (4 of them solved in the root node of the B&C tree). For 23 instances we could not even find a feasible solution, such state denoted by "–". Moreover, we could not solve 36 instances to optimality within the time limit, although feasible solutions for them were found. We can infer from the computational test results that the larger the instance parameter values are, the larger the problem is and the most difficult it is to find solutions for it. With high 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 even further from the optimal solution.

Table 32 – 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	1,394	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	2,455	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	3,240	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	1,112	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	2,422	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	2,711	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	3,554	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	6,188	0.98%	3600.0	20.81	5.37	cwp057	43.15	–	–	–	3600.0	28.48	135.68
cwp023	31.50	35.20	6,342	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	7,495	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	7,226	5.92%	3600.0	15.03	5.03	cwp062	76.40	–	–	–	3600.0	40.02	469.50
cwp028	25.40	26.30	9,273	8.06%	3600.0	18.15	6.43	cwp063	108.55	–	–	–	3600.0	53.72	103.78
cwp029	23.50	25.80	7,529	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	8,024	1.09%	3600.0	18.56	4.12	cwp065	86.60	–	–	–	3600.0	45.36	624.61
cwp031	42.80	47.00	1,530	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	3,600	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	2,102	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

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 17.

Figure 17 – Objective function values for integer model solutions, linear relaxation solutions and proposed lower bound value for test instances



In Figure 17, 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.

3.6.3 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. To cite an example, Gholami *et al.* (2009) used Taguchi experimental design (JR, 1988) to achieve improved robustness of the genetic algorithm which they proposed. In this method the optimal parameter choice if 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. We can see in Table 33 the proposed levels for the genetic algorithm parameters (control factors) introduced in Section 3.5.

Table 33 – 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 a 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 we may use the *D-optimal* design(AGUIAR *et al.*, 1995), which are constructed to minimize 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 (TRIEFENBACH, 2008) using R programming language for 9 trials for the chosen factors and their respective levels, illustrated in Table 34.

Table 34 – *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

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.

$$\text{S/N ratio:} \quad \eta_i = -10 \ln \left(\frac{1}{N} \sum_{j=1}^{N} FIT_{ij} \right), \tag{3.35}$$

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:

$$LBD_{ij} = \frac{FIT_{ij} - \mathrm{LB}_j}{\mathrm{LB}_j}, \tag{3.36}$$

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:

$$LBD_i = \frac{1}{N} \cdot \sum_{j=1}^{N} LBD_{ij}, \tag{3.37}$$

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.

We can see in Table 35 the results after carrying out the computational tests for each trial with the test instance set.

Table 35 – 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

In Figure 18, we can see the main effects plot for the control factors using S/N ratios as the response variable. In Figure 19, 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.

Figure 18 – Main effects plot for S/N ratio for lowerbound deviation values

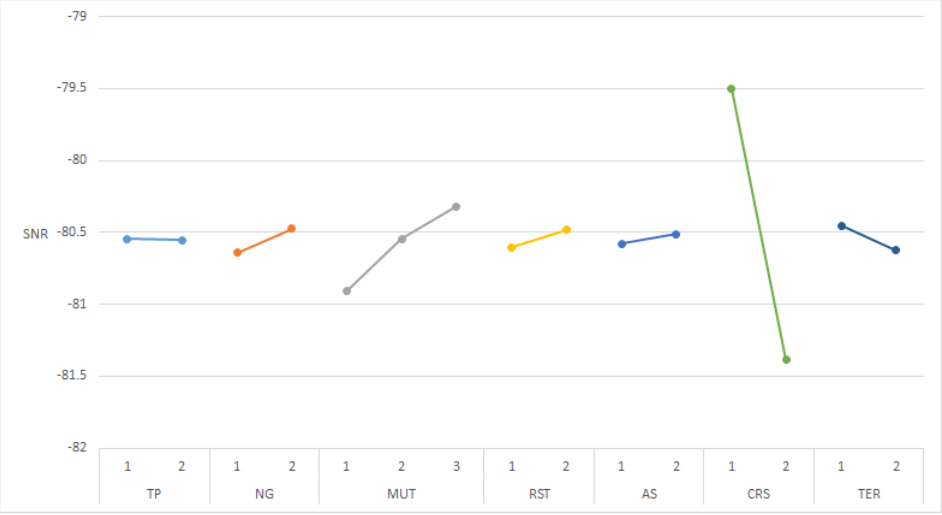
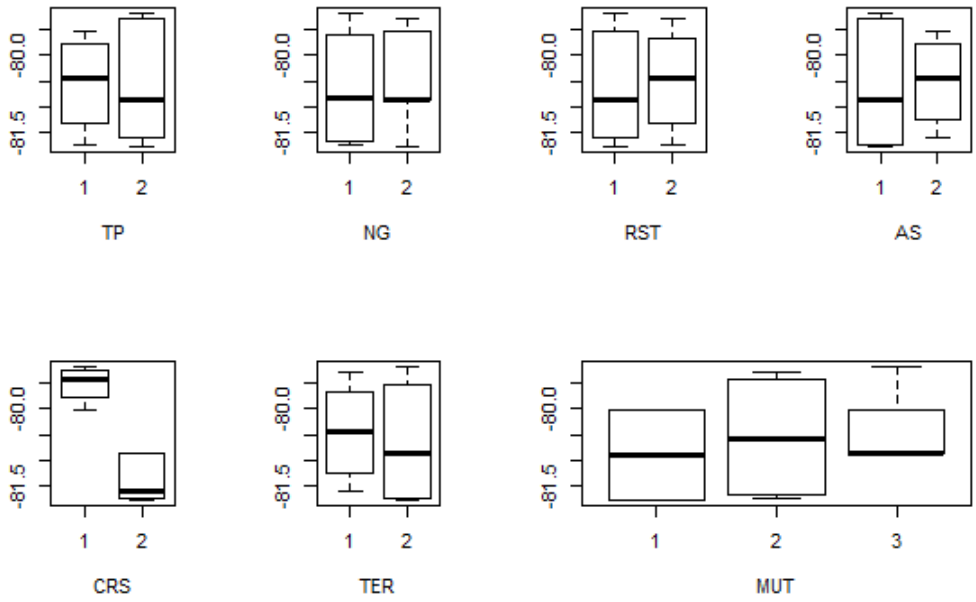


Figure 19 – Boxplots for S/N ratio values with each factor



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 37.

Table 36 – 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 37 – 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	

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 20 and in boxplots in Figure 21 that LBD have a similar behavior on the control factors. However, we note that, in this case, the lower the LBD value the better.

Figure 20 – Main effects plot for lowerbound deviation

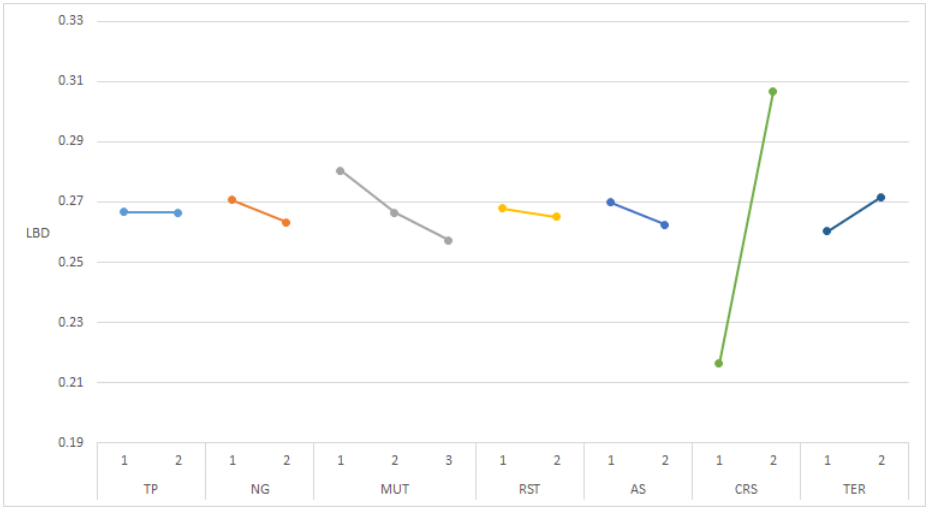
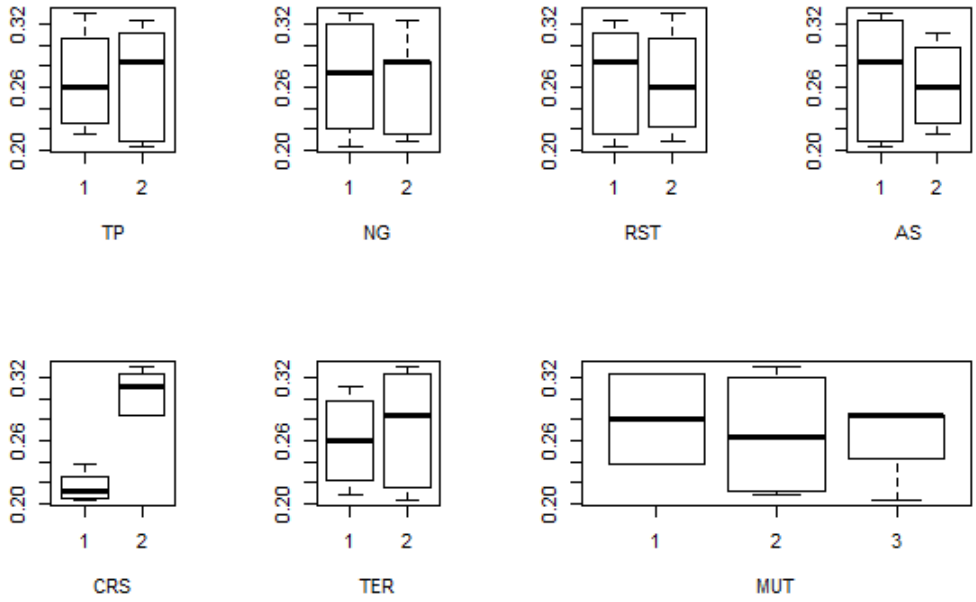


Figure 21 – Boxplots for LBD values with each factor



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 37.

Table 38 – 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 39 – ANOVA table for LBD values for linear regression model fit considering most significant factors

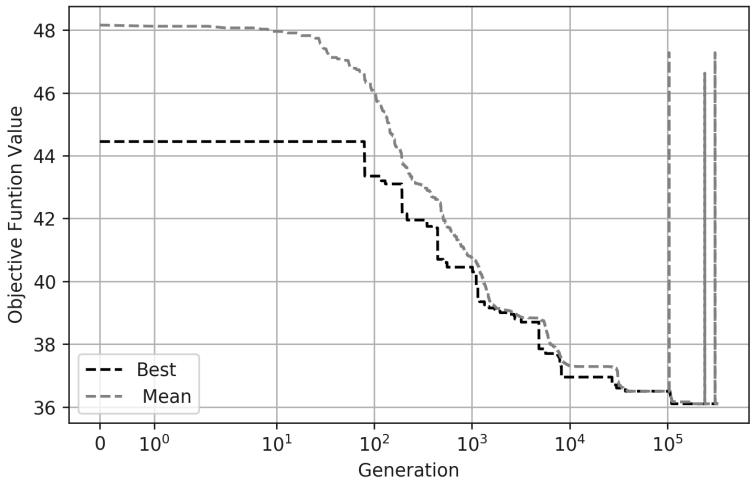
Factors	**df**	**Sum Sq**	**Mean Sq**	***F* value**	***P*-value**	
MUT	1	0.00063	0.00063	6.02770	0.04944	*
CRS	1	0.01949	0.01949	187.86960	0.00001	***
Residuals	6	0.00062	0.00010			
Total	8	0.02074				
Signif. codes:		'***' 0	'*' 0.05			

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

3.6.4 Analysis of the final genetic algorithm parameterization

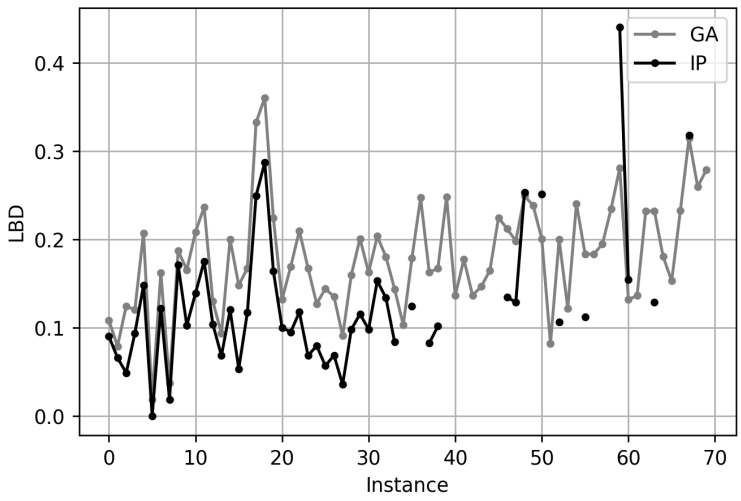
In order to observe the genetic algorithm behaviour, we run the GA with instance cwp021. Figure 22 illustrates the best fitness and mean fitness of the populations along all generations. The *x-axis* of the Figure 22 is on logarithmic scale. The largest improvement takes place during the first generations of the GA, while in the last ones the best fitness is stagnant with some improvement upon the first restart.

Figure 22 – Average and best objective function value curves for instance cwp021 along generations of the selected genetic algorithm parameterization



In Figure 23, 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.

Figure 23 – Lower bound relative deviations for CPLEX and GA with the selected parameterization



In Figures 24 and 25, 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* 25 is in logarithmic scale.

Figure 24 – Mean time for each instance solved by CPLEX and GA with the selected parameterization

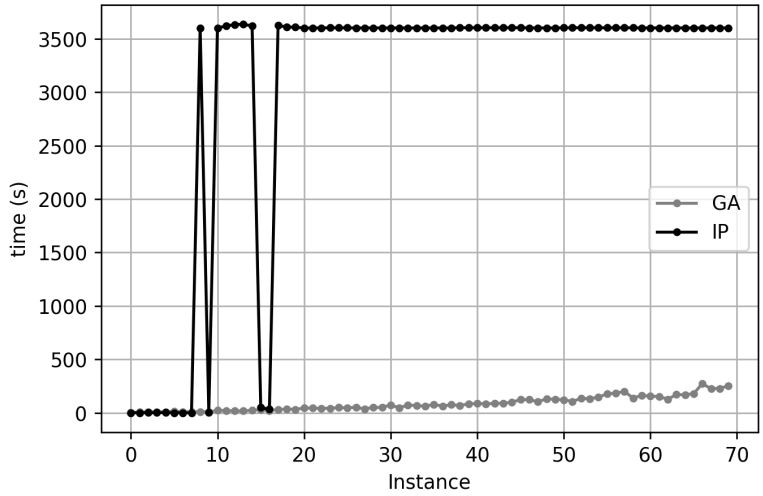
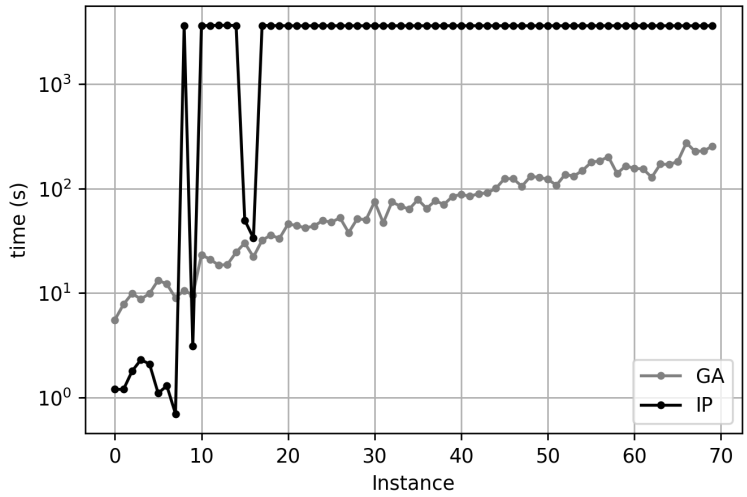


Figure 25 – Mean time for each instance solved by CPLEX and GA with the selected parameterization, with y-axis in logarithmic scale



3.7 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*, which, to the best of the authors' knowledge, have not yet been studied and may have a large impact on both real-world and theoretical studies. The proposed variant consists in integrating the problem of production planning of precast beam with the problem of cutting the traction elements used in such production, while taking into consideration the generation of leftovers and

bar generated via overlapping. We argued that such variant is NP-Hard and proposed an integer linear programming model as well as a lower bound for its optimal objective function value. We also argued that restricting the formulation to using exclusively maximal packing patterns does not change the optimal solution set of the proposed model.

We proposed a constraint programming model for generating each type of pattern, total of three models. We suggested a benchmark instance set for carrying out computational experiments for the evaluation of the solution methods. The integer programming model can be used to solve small size instances, while it typically does not reach optimality while solving medium size instances. In addition, the model usually does not find feasible solutions for large size instances. We introduced a genetic algorithm for solving the problem and we performed a *D-optimal* experimental design to achieve improved robustness of the algorithm. Then, we compared CPLEX results for the integer programming model with the final genetic algorithm parameterization obtained vie the experimental design. We can infer from the computational tests that the proposed genetic algorithm is an attractive alternative to the integer programming model resulting in high-quality solutions in a quick solution time as compared with the exact model.

There are numerous opportunities for future work regarding the ICP-HPBMPP. To give some examples in the domain of modeling, the problem can be adapted to consider distinct different 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 of time we would have a new beam demand to be fulfilled, and a limited stock of bars. Regarding the 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.

4 CONCLUSIONS

In this master thesis, we introduced two novel variants of cutting and sequencing problems. We proposed integer linear programming models and heuristic approaches for their solution. Among the solution methods that we studied, we highlight the size-reduction heuristic for the HPPBMPP, which allowed us to obtain several high-quality solutions within short solution times and with modest memory usage when compared to the integer linear models using maximal patterns. The proposed genetic algorithm for the ICP-HPBMPP was able to produce near-optimal solutions in a fraction of the time requires to solve the exact model via CPLEX. The genetic algorithm was also able to produce high-quality solutions for instances for which we could not find feasible solutions using the exact approach.

The use of constraint programming (CP) for generating patterns the test instances for both variants proved to be an excellent approach for dealing with the vast number of combinations of such patterns. The CP models are highly flexible and spawn several solutions in an almost negligible computational time. We explored such flexibility with introduction of maximal patterns, resulting in a large decrease of the size of the proposed models, while maintaining their optimal solutions.

There are several opportunities for future work in the variants of the problems studied in this work. In this thesis we only focused on few areas. The proposed integer linear models are suitable for decomposition approaches, such as column generation which is a frequently used technique for solving cutting stock problems. Different modeling strategies can also be studied and compared to the two problems. Future studies can address models that are more adherent to real-world problems with respect to, for example, the use of as additional scheduling constraints, dynamic demands, delivery dates, and dynamic use of leftover material. In addition, when it comes to the method used for parameterization of the genetic algorithm, experimental designs considering more factor levels and/or interaction effects among factors may be devised.

BIBLIOGRAPHY

AGUIAR, P. F. de; BOURGUIGNON, B.; KHOTS, M.; MASSART, D.; PHAN-THAN-LUU, R. D-optimal designs. **Chemometrics and intelligent laboratory systems**, Elsevier, v. 30, n. 2, p. 199–210, 1995.

ARAUJO, K.; BONATES, T.; PRATA, B.; PITOMBEIRA-NETO, A. **Heterogeneous Prestressed Precast Beams Multiperiod Production Planning Problem: Modeling and Solution Methods**. 2019. Preprint, <https://arxiv.org/abs/1903.08609v1>.

ARBIB, C.; MARINELLI, F. On cutting stock with due dates. **Omega**, Elsevier, v. 46, p. 11–20, 2014.

ARENALES, M. N.; CHERRI, A. C.; NASCIMENTO, D. N. d.; VIANNA, A. A new mathematical model for the cutting stock/leftover problem. **Pesquisa Operacional**, SciELO Brasil, v. 35, n. 3, p. 509–522, 2015.

BELDICEANU, N.; CARLSSON, M. **Global Constraint Catalog**. 2018. Available from Internet: <http://sofdem.github.io/gccat/gccat/Celement.html>.

BENJAORAN, V.; BHOKHA, S. Three-step solutions for cutting stock problem of construction steel bars. **KSCE Journal of Civil Engineering**, Springer, v. 18, n. 5, p. 1239–1247, 2014.

BENJAORAN, V.; DAWOOD, N.; HOBBS, B. Flowshop scheduling model for bespoke precast concrete production planning. **Construction Management and Economics**, Taylor & Francis, v. 23, n. 1, p. 93–105, 2005.

BRAGA, N.; ALVES, C.; DE CARVALHO, J. V. Exact solution of combined cutting stock and scheduling problems. In: **Computational Management Science**. [S.l.]: Springer, 2016. p. 131–139.

CHEN, J.-H.; YAN, S.; TAI, H.-W.; CHANG, C.-Y. Optimizing profit and logistics for precast concrete production. **Canadian Journal of Civil Engineering**, NRC Research Press, v. 44, n. 999, p. 393–406, 2017.

DE CASTILHO, V. C.; EL DEBS, M. K.; DO CARMO NICOLETTI, M. Using a modified genetic algorithm to minimize the production costs for slabs of precast prestressed concrete joists. **Engineering applications of artificial intelligence**, Elsevier, v. 20, n. 4, p. 519–530, 2007.

DYCKHOFF, H. A typology of cutting and packing problems. **European Journal of Operational Research**, Elsevier, v. 44, n. 2, p. 145–159, 1990.

FANJUL-PEYRO, L.; PEREA, F.; RUIZ, R. Models and matheuristics for the unrelated parallel machine scheduling problem with additional resources. **European Journal of Operational Research**, Elsevier, v. 260, n. 2, p. 482–493, 2017.

FANJUL-PEYRO, L.; RUIZ, R. Size-reduction heuristics for the unrelated parallel machines scheduling problem. **Computers & Operations Research**, Elsevier, v. 38, n. 1, p. 301–309, 2011.

GAREY, M.; JOHNSON, D.; COLLECTION, M. S. M. **Computers and Intractability: A Guide to the Theory of NP-completeness**. W. H. Freeman, 1979. (Books in mathematical series). ISBN 9780716710448. Available from Internet: <https://books.google.com.br/books?id=fjxGAQAAIAAJ>.

GHOLAMI, M.; ZANDIEH, M.; ALEM-TABRIZ, A. Scheduling hybrid flow shop with sequence-dependent setup times and machines with random breakdowns. **The International Journal of Advanced Manufacturing Technology**, Springer, v. 42, n. 1-2, p. 189–201, 2009.

GILMORE, P. C.; GOMORY, R. E. A linear programming approach to the cutting-stock problem. **Operations research**, INFORMS, v. 9, n. 6, p. 849–859, 1961.

GILMORE, P. C.; GOMORY, R. E. A linear programming approach to the cutting stock problem—part ii. **Operations research**, INFORMS, v. 11, n. 6, p. 863–888, 1963.

GRAMANI, M. C. N.; FRANÇA, P. M. The combined cutting stock and lot-sizing problem in industrial processes. **European Journal of Operational Research**, Elsevier, v. 174, n. 1, p. 509–521, 2006.

JR, J. J. P. An overview of the strategy and tactics of taguchi. **IIE transactions**, Taylor & Francis, v. 20, n. 3, p. 247–254, 1988.

KHALILI, A.; CHUA, D. Integrated prefabrication configuration and component grouping for resource optimization of precast production. **Journal of Construction Engineering and Management**, American Society of Civil Engineers, v. 140, n. 2, p. 1–12, 2013.

KO, C.-H.; WANG, S.-F. Precast production scheduling using multi-objective genetic algorithms. **Expert Systems with Applications**, Elsevier, v. 38, n. 7, p. 8293–8302, 2011.

LI, S. Multi-job cutting stock problem with due dates and release dates. **Journal of the Operational Research Society**, Springer, v. 47, n. 4, p. 490–510, 1996.

MARLER, R. T.; ARORA, J. S. Survey of multi-objective optimization methods for engineering. **Structural and multidisciplinary optimization**, Springer, v. 26, n. 6, p. 369–395, 2004.

MELEGA, G. M.; ARAUJO, S. A. de; JANS, R. Classification and literature review of integrated lot-sizing and cutting stock problems. **European Journal of Operational Research**, Elsevier, 2018.

NONÅS, S. L.; THORSTENSON, A. A combined cutting-stock and lot-sizing problem. **European Journal of Operational Research**, Elsevier, v. 120, n. 2, p. 327–342, 2000.

NONAS, S. L.; THORSTENSON, A. Solving a combined cutting-stock and lot-sizing problem with a column generating procedure. **Computers & Operations Research**, Elsevier, v. 35, n. 10, p. 3371–3392, 2008.

PILEGGI, G. C.; MORABITO, R.; ARENALES, M. N. Abordagens para otimização integrada dos problemas de geração e seqüenciamento de padrões de corte: caso unidimensional. **Pesquisa Operacional**, SciELO Brasil, v. 25, n. 3, p. 417–447, 2005.

POLDI, K. C.; ARENALES, M. N. O problema de corte de estoque unidimensional multiperíodo. **Pesquisa Operacional**, SciELO Brasil, v. 30, n. 1, p. 153–174, 2010.

PRATA, B. A.; PITOMBEIRA NETO, A. R.; SALES, C. J. M. An integer linear programming model for the multiperiod production planning of precast concrete beams. **Journal of Construction Engineering and Management**, American Society of Civil Engineers, v. 141, n. 10, p. 1–4, 2015.

REINERTSEN, H.; VOSSEN, T. W. The one-dimensional cutting stock problem with due dates. **European Journal of Operational Research**, Elsevier, v. 201, n. 3, p. 701–711, 2010.

SALEM, O.; SHAHIN, A.; KHALIFA, Y. Minimizing cutting wastes of reinforcement steel bars using genetic algorithms and integer programming models. **Journal of Construction Engineering and Management**, American Society of Civil Engineers, v. 133, n. 12, p. 982–992, 2007.

SCHULTE, C.; TACK, G.; LAGERKVIST, M. Z. **Modeling**. 2016. Corresponds to *Modeling and Programming with Gecode 5.0.0*.

SHAHIN, A. A.; SALEM, O. M. Using genetic algorithms in solving the one-dimensional cutting stock problem in the construction industry. **Canadian journal of civil engineering**, NRC Research Press, v. 31, n. 2, p. 321–332, 2004.

SHIH, K.-C.; LIU, S.-S. An optimization model for precast project planning using group concepts. **Journal of the Operations Research Society of Japan**, , v. 53, n. 3, p. 189–206, 2010.

STADTLER, H. A one-dimensional cutting stock problem in the aluminium industry and its solution. **European Journal of Operational Research**, Elsevier, v. 44, n. 2, p. 209–223, 1990.

TRIEFENBACH, F. Design of experiments: the d-optimal approach and its implementation as a computer algorithm. **Bachelor's Thesis in Information and Communication Technology**, Citeseer, 2008.

TRKMAN, P.; GRADISAR, M. One-dimensional cutting stock optimization in consecutive time periods. **European Journal of Operational Research**, Elsevier, v. 179, n. 2, p. 291–301, 2007.

VANCE, P. H. Branch-and-price algorithms for the one-dimensional cutting stock problem. **Computational Optimization and Applications**, Springer, v. 9, n. 3, p. 211–228, 1998.

VASSOLER, A. H. D.; POLTRONIERE, S. C.; ARAUJO, S. A. Modelagem matemática para o problema de produção de vigotas na indústria de lajes treliçadas. **Revista Eletrônica Paulista de Matemática**, v. 7, p. 68–77, dec 2016.

WANG, Z.; HU, H.; GONG, J. Framework for modeling operational uncertainty to optimize offsite production scheduling of precast components. **Automation in Construction**, Elsevier, v. 86, p. 69–80, 2018.

WÄSCHER, G.; GAU, T. Heuristics for the integer one-dimensional cutting stock problem: A computational study. **Operations-Research-Spektrum**, v. 18, n. 3, p. 131–144, 1996. ISSN 1436-6304.

WÄSCHER, G.; HAUSSNER, H.; SCHUMANN, H. An improved typology of cutting and packing problems. **European Journal of Operational Research**, Elsevier, v. 183, n. 3, p. 1109–1130, 2007.

YANASSE, H. H.; LAMOSA, M. J. P. An integrated cutting stock and sequencing problem. **European Journal of Operational Research**, Elsevier, v. 183, n. 3, p. 1353–1370, 2007.

YANG, Z.; MA, Z.; WU, S. Optimized flowshop scheduling of multiple production lines for precast production. **Automation in Construction**, Elsevier, v. 72, p. 321–329, 2016.

YUEN, B. J. Heuristics for sequencing cutting patterns. **European Journal of Operational Research**, Elsevier, v. 55, n. 2, p. 183–190, 1991.

APPENDIX A – LARGE ALGORITHMS FROM CHAPTER 3

Consider *frequency(Pattern)* as the frequency associated with Pattern in its respective gene, and the notation "#" standing for expression "number of" for the following algorithms.

Algorithm 4: Remove unnecessary packing patterns

input: Infeasible chromosome
output: 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 5: Fix chromosome with respect to infeasibility 1

input: Infeasible chromosome
output: 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
```

Algorithm 6: Fix chromosome with respect to infeasibility 2

input: Infeasible chromosome
output: 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 ← #bars *w* used - stock of *w* bars;
      frequency(I_h) ← 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 ← #bars *w* used - stock of *w* bars;
      rt ← $\left\lfloor \frac{rt}{\text{\#bars w in } O_\mu} \right\rfloor$
      frequency(O_μ) ← 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*

Algorithm 7: Fix chromosome with respect to infeasibility 3

input: Infeasible chromosome
output: 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 $\leftarrow \left\lceil \frac{rt}{\text{\#bars } \gamma \text{ generated by } I_h} \right\rceil$
      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 $\leftarrow \left\lfloor \frac{rt}{\text{\#bars } \gamma \text{ generated by } I_h} \right\rfloor$
      frequency(I_h) frequency $\leftarrow$ 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*