Scheduling of Semi-Automatic Carousels with Fixed Production Sequences

Giampaolo Campana[a], Enrico Malaguti[b], Mattia Mele[a], Paolo Paronuzzi[b]

[a]DIN, Università di Bologna, Bologna, Italy; [b]DEI "Guglielmo Marconi", Università di Bologna, Bologna, Italy

ARTICLE HISTORY
Compiled 2024-05-09

ABSTRACT
Rotary transfer machines are widely used in different industrial sectors. A rich literature concerning their design and optimisation is available, but mainly dedicated to integrated machining systems. This machine architecture is also implemented in the aluminum gravity die casting technology where the specificity of the casting process needs an appropriate design. In particular, the constraints related to processing times and the rigidity of the production sequence impose a specific approach to schedule the production for achieving an optimal cycle time.

We approach this problem with an optimisation perspective: first we propose a mixed-integer linear programming formulation for defining the sequencing and scheduling of the machine in order to obtain a specified production with minimum makespan, and discuss strategies for enumerating the variables of the formulation. Second, we describe a heuristic algorithm as an alternative to the solution of the formulation through a general-purpose solver.

Eventually, we present extensive computational experiments on a set of instances generated from real data, comparing these alternative approaches.

KEYWORDS
Rotary Transfer Machine; Gravity Die Casting; Scheduling; Flow-shop; Cutting-Stock; Setups; Mixed-integer Linear Program, Heuristics.

1. Introduction

Transfer machines with rotary table are widely used in different industrial sectors because a high automation and efficiency can be achieved by partitioning operations into groups, which are then performed in designed sequences and in fixed working positions (Battaia et al. 2012). In particular, Rotary Transfer Machines (RTMs) consist in a table comprising a number of fixed positions where parts are located and machined by spindle head or turrets. Through discrete rotations of the table around its central axis, each part is moved to the different working stations where all the scheduled manufacturing processes take place (Richter and Ondrášek 2017). RTMs are fully automated systems that aim at standardising manufacturing processes while minimising idle times between consecutive operations.

A similar machine architecture is dedicated to foundry processes. Particularly, Semi-Automatic Carousels (SAC) are used in aluminum Gravity Die Casting (GDC) because

CONTACT Enrico Malaguti. Email: enrico.malaguti@unibo.it

they facilitate the semi-automatic handling of dies and allow for an efficient use of the space. This casting process owns specific characteristics and needs a customised approach to define the number of working positions and the carousel cycle time (Campana, Mele, and Cimatti 2020).

The manufacturing cycle of a product by GDC comprises four different phases, namely (i) mould preparation, (ii) metal pouring (filling), (iii) part solidification by cooling, and (iv) solid part extraction. Once a part has been extracted, the mould is re-prepared for casting a new part. This cyclic nature makes GDC ideal for carousel. In fact, once a mould has been positioned on the rotating table, it can be used to manufacture several parts without further handling. Specifically, we denote as M the maximum number of manufacturing cycles that a generic j-th mould can perform on the carousel. This number depends on the wearing of the mould painting occurring during metal pouring and part extraction. Therefore, after M cycles, the mould must be took off from the carousel and refurbished before being reused.

An example of carousel for GDC comprising s stations is schematically shown in Fig. 1 a). The carousel carries s dies, which are transferred through the different stations by discrete rotations with an angular step equal to $\frac{2\pi}{s}$.

Fig. 1 b) illustrates the phases of casting for a mould going through the carousel. In station 1, dies are opened and empty. An operator prepares and verifies the mould in the first operation of the casting cycle. In the next station 2, prepared moulds are filled with the molten metallic alloy. Once moulds are filled, the metal starts cooling down and solidifies while the die keeps moving between stations 3 and $(s-1)$. Finally, the solid part is extracted from the mould in the last station s of the production cycle. The mould preparation in station 1 is performed manually as it includes the handling of fragile cores. Metal casting and part extraction in station 2 and S, respectively, are carried out by robots. The movements of the carousel are governed by a Programmable Logic Controller (PLC). This fact means that it is possible to set up a sequence of different time spans between consecutive rotations to allow for the completion of each operation and minimisation of the overall cycle time. If security issues occur, the automated rotation of the carousel can be stopped by the operator in station 1.

It is worth underlying that the duration of each phase of the manufacturing cycle is different for each mould assembled on the carousel because it depends on the shapes and size of manufactured products. Specifically, the preparation time τ_j^p must allow the operator to properly restore the mould. The filling time of the mould τ_j^c is a major concern in GDC. In fact, high pouring speed might determine turbulence in the metal flow triggering defects in final parts. On the other hand, slow pouring speeds mean that the liquid metal cools down and possibly solidifies in some regions of the cavity leading to an incomplete filling. Thus, the proper pouring time must be determined during the process design phase, usually by means of Finite Element Analysis (FEA) (Ha et al. 1999; Cleary et al. 2002).

FEA of the process is also used to calculate the solidification time τ_j^s of casting products, which depends on the heat exchange coefficient between the metal and the die, the mould cavity geometry and other factors, such as the use of chillers (Vijayaram et al. 2006; Kamal et al. 2018). The solidification phase by cooling has a lower and a upper time constraint. The solidification of the entire geometry must be prolonged until the achievement of that level of stiffness necessary to avoid part deformation during extraction (lower time constraint). Particularly, after the filling is completed, the metal must remain in the mould for a minimum time τ_j^{sl}. On the contrary, the solidification cannot be too long: the solid part must be extracted at a sufficiently high temperature, otherwise the metal adheres to the mould cavity due to shrinkage

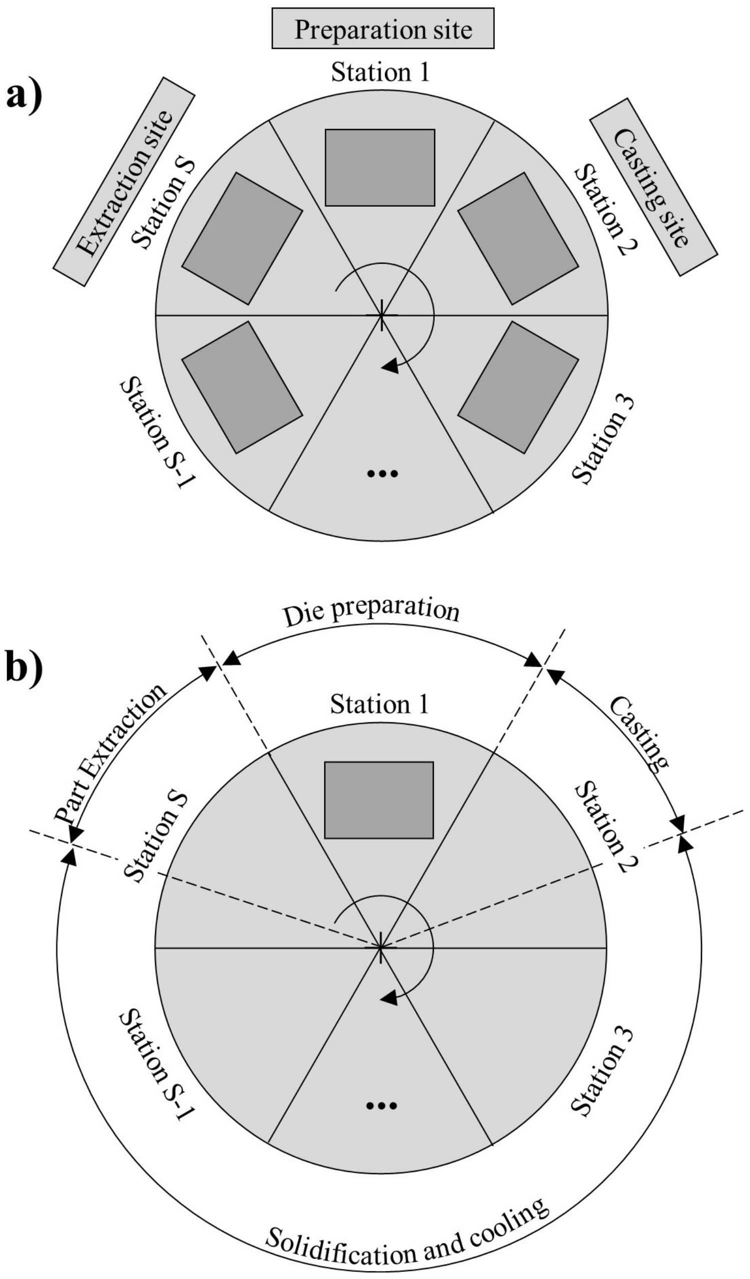

Figure 1. Caption: Scheme of the rotary machine.
Alt text: A carousel carrying s dies, and an illustration of the phases of casting for a mould going through the carousel.

phenomena during its cooling by impeding its final removal (upper time constraint). Thus, to prevent this phenomenon, it is necessary to define a maximum time τ_j^{su} for the solidification and cooling phase. Finally, the extraction time τ_j^e mainly depends on the geometry of the mould cavity.

Based on the previous considerations, it is possible to define, for the generic j-th mould, a vector τ_j comprising the characteristic cycle times, as shown in (1).

$$\tau_j = [\tau_j^p, \tau_j^c, \tau_j^e, \tau_j^{sl}, \tau_j^{su}] \tag{1}$$

The values in (1) only depend on the product geometry. As well known, in GDC it is frequent to have more than one mould for each product so as to increase the productivity. In this case, we denote as Q_j the number of moulds, and the vector τ_j is identical for all the moulds that are identical.

The rotations of the carousel must consider that the time spent by the part in each station is compatible with the manufacturing needs expressed by τ_j. Since a rotation of the carousel determines the simultaneous transfer of all the moulds to the next station, the minimum time steps are strictly related to the order of moulds that are assembled on the carousel. Therefore, the optimisation of the carousel consists in the definition of the optimal sequence of products and the related time steps for each rotation. It is worth underlying that, unlike in the case of RTM for machining (Afrin et al. 2016; Battaïa, Dolgui, and Guschinsky 2017b), the problem presents an upper constraint to the time cycle. This in turn limits the maximum number of stations in the carousel.

In a typical industrial scenario, each product is characterised by a demand U_j over a certain time span, e.g. one month. Different sets of moulds are loaded on the carousel in subsequent production phases, and the associated sequences are replicated a number of times in order to satisfy the product demands. Changing sequence on the carousel requires a setup whose time includes a minimum warm-up time τ_j^W associated with each product in the sequence, and the time for the first rotation of the platform (during which no product is obtained). Since setting-up a sequence on the carousel is a labour-intensive and time-consuming operation, it is quite common in the industry to have constraints imposing to perform at least a minimum number μ of replications for each sequence. Overall, a scheduling problem of the subsequent sequences arises in the industrial use of RTM in Aluminum GDC, besides the sequencing one mentioned above. In this paper we consider the following optimisation problem:

Problem statement: Scheduling of a Semi-Automatic carousel. We are given a set of N different products to be realised and, for each product $j = 1, \ldots, N$, the cycle times τ_j as in (1), the mould setup time τ_j^W, the availability of moulds Q_j, and the demand U_j. We are also given the maximum number of times M a sequence can be replicated before the moulds are refurbished, as well as the minimum number of replications μ for a sequence. We want to define: i) the optimal sequences of products to be processed on the RTM; ii) the duration of each step in the defined sequences; iii) the order in which sequences are loaded on the RTM, so that a mould that needs to be refurbished is not used in a subsequent production phase; and iv) the number of times each sequence is replicated, in such a way that all the demand is satisfied with minimum makespan.

Numerical example. We report a numerical example with 10 products, denoted as A, B, …, L, and 6 stations. For each product, Table 1 reports the cycle times and the demand ($\tau_j^{sl} = 15$, $\tau_j^{su} = 30$ and $\tau_j^W = 100$ for all the products). Table 2 represents an optimal solution obtained by defining two sequences, reporting for each one: the index of the sequence p; the products in the sequence and the order in which they are processed on the RTM; the optimal number of replications λ_p; the sequence cycle time t_p (defined as $t_p = \sum_{t=1\ldots6} d_t$, where d_t is the duration of step t) and setup time

W_p, this last figure is obtained by summing to the cycle time t_p the largest mould setup time τ_j^W (equal to 100 for all products), i.e., $W_p = t_p + \max_{j \in p} \tau_j^W$. In this example the order in which the sequences are scheduled on the RTM is irrelevant. In this solution, products A and B are overproduced, as their demands are equal to 9 and 8, respectively, while the actual production is 10 for both. Finally, Table 3 describes the configuration of the RTM during the 6 steps of sequence [B,C,H,G,D,L]. For each step, the table reports the station occupied by each product, the time required for the operations of preparation, casting and extraction, where we highlighted in bold the operation(s) determining the duration of the step (corresponding to the value of row d_t). In this case, we always have $d_t = \max\{\tau_{j_1}^p, \tau_{j_2}^c, \tau_{j_3}^e\}$, where we indicate with j_1, j_2, j_3 the product in the Preparation, Casting and Extraction station, respectively. In parenthesis, for products that are in a solidification and cooling station, the table reports the cumulative solidification and cooling time, i.e., the overall time spent by the product in solidification and cooling at the end of that step. This sequence allows to extract each product after at least $\tau_j^{sl} = 15$ seconds and no more than $\tau_j^{su} = 30$ seconds of cooling.

Table 1. Cycle times and demand for the numerical example.

	τ_j^p	τ_j^c	τ_j^e	U_j
A	7	6	7	9
B	3	10	9	8
C	8	3	2	10
D	4	8	1	20
E	5	5	1	20
F	7	6	5	10
G	10	5	4	10
H	4	10	4	10
I	8	5	7	10
L	9	3	1	10

In spite of the widespread use of carousels in the GDC industry, to the best of authors' knowledge this problem has not been previously investigated in the literature. The reminder of this section reviews the relevant optimisation literature. In Section 2, we propose a Mixed-Integer Linear Programming (MILP) formulation for the problem, and discuss strategies for enumerating its variables. In Section 3 we describe a heuristic algorithm as an alternative to the solution of the MILP formulation through a general-purpose solver. Extensive computational experiments comparing these alternative approaches are presented in Section 4.

Table 2. Optimal solution for the numerical example.

p	products	λ_p	t_p	W_p
1	[B,C,H,G,D,L]	10	41	141
2	[A,I,E,E,D,F]	10	38	138

Table 3. Situation on the RTM during the 6 steps of sequence [B,C,H,G,D,L].

	Step 1		Step 2		Step 3		Step 4		Step 5		Step 6	
Preparation	**3**	**B**	8	C	**4**	**H**	**10**	**G**	4	D	**9**	**L**
Casting	**3**	**L**	**10**	**B**	3	C	**10**	**H**	**5**	**G**	8	D
Cooling	(3)	D	(10)	L	(4)	B	(10)	C	(5)	H	(9)	G
Cooling	(12)	G	(13)	D	(14)	L	(14)	B	(15)	C	(14)	H
Cooling	(17)	H	(22)	G	(17)	D	(24)	L	(19)	B	(24)	C
Extraction	2	C	4	H	**4**	**G**	1	D	1	L	**9**	**B**
d_t	3		10		4		10		5		9	

1.1. *Literature review*

The scope of our review of the literature is twofold: on the one hand, we position our problem within the optimisation literature; on the other hand, we survey related optimisation problems on RTMs.

Problem positioning: scheduling and cutting-stock. By leaving behind the fact that the transfer is performed with a rotary platform carrying the moulds (and hence the sequence of processed products repeats identically until a setup is performed), this scheduling problem can be interpreted as a variant of the flow shop-sequencing problem with synchronous transfers, defined by Soylu, Kirca, and Azizoğlu (2007). However, the periodic repetition of the sequence of products is the key feature of the problem and hence of its optimisation. After defining the possible sequences of products, which, in addition to the processing order, asks to define the timing of each step, we still have to decide which sequences we want to load on the RTM and how many times to replicate each sequence. This resulting problem also has a natural interpretation as a variant of the cutting-stock problem with setups, involving additional complicating constraints. Indeed, in a cutting-stock interpretation, a sequence is equivalent to a pattern, the products in the sequence are the items in the pattern, its processing time (intended as the time for a full rotation of the RTM) is its cost, and we want to decide the number of times each sequence is replicated (each pattern is cut) in order to satisfy the demand at minimum cost: this is exactly the cutting-stock problem (see, e.g., Martello and Toth (1990)). The cutting-stock interpretation of the problem at study immediately allows us to describe its core according to the classical formulation due to Gilmore and Gomory (1963). The problem is further complicated by setups that occur each time a different sequence (pattern) is processed on the RTM; once a sequence is selected, industrial considerations impose that it is replicated a minimum number of times (see also Bo et al. (2021) on minimum production constraints); and after a maximum number of replications of the same sequence, a new setup takes place. Finally, the processing order of sequences must be defined, because it is constrained by the availability of moulds. Indeed, an exploited mould must be refurbished and cannot be used in two sequences that are processed one after the other.

In our setting, the minimisation of the makespan is obtained by minimising the sum of the actual processing time and setup time. This second aspect is strictly related with the problem of minimising the number of different patterns in a cutting-stock problem, when each change of pattern requires a setup. The *cutting-stock problem with pattern minimisation* was initially addressed by Haessler (1975), who proposed a heuristic algorithm where the cost of using a pattern is added as an additional cost term to the objective function. Later, Foerster and Wascher (2000) proposed a method for minimising the number of different cutting patterns once the amount of stock is determined.

Belov and Scheithauer (2006) proposed a sequential heuristic for the combined minimisation of the stock and of the number of different cutting patterns. Concerning exact approaches, Vanderbeck (2000) presented an MILP formulation and a branch-and-cut-and-price algorithm for minimising the number of different patterns, once the maximum number of stock elements is fixed as a constraint (notice that in the classical cutting-stock each stock element has the same cost).

In addition to setups that occur when changing production sequence, we have additional setups occurring when the number of replications of a sequence exceeds the moulds life, and a new set of moulds must be loaded on the RTM, while exploited moulds are refurbished. By considering the cost of the setups of a specific sequence as a function of the number of times the sequence is replicated, a staircase function results, with steps occurring at each multiple of the moulds life. A similar cost function is encountered in the wood-cutting industry, where a given number of wood panels can be stacked and processed in a single cutting cycle when they have the same pattern, thus reducing the number of cycles (equivalent to the number of setups in our setting). Malaguti, Durán, and Toth (2014) formulated this related cutting-stock problem and tackled it through a truncated branch-and-price algorithm. Later (Malaguti, Durán, and Toth 2016), they proposed a metaheuristic algorithm for solving optimisation problems where the objective function has a staircase component, which partially inspired the metaheuristic algorithm we describe in this paper.

Design and planning of RTMs. Concerning specifically RTMs (not restricted to the foundry industry), a rich literature exists on their design for parallel machining of multiple parts in batches. This is a complex problem due to the large number of possible and alternative architectures. The design of these machines needs a high number of technical decisions based on both manufacturing and product design constraints but it also depends on managerial or economic choices (Battaïa et al. 2013). Despite this complexity, a certain number of machines are available on the market and, consequently, several methods are available to optimise the setting of each working station, based on the solution of the line balancing problems. Heuristic techniques are mainly used for this purpose by sequentially assigning all the operations to machining modules while minimising the equipment cost (Battaïa, Dolgui, and Guschinsky 2016). Concerning optimisation models, Dolgui, Guschinsky, and Levin (2009) and Battaïa et al. (2014) presented mixed-integer programming formulations to minimise the cost of a rotary production line, while achieving the desired cycle time, subject to technological and precedence constraints. An alternative approach based on the reduction of the problem to a constrained shortest path was also presented. Battaïa et al. (2012) and Battaïa, Dolgui, and Guschinsky (2017a) presented similar MIP models for designing reconfigurable rotary machinines; also in these cases the problem was solved by means of heuristic procedures.

The same authors (Battaïa, Dolgui, and Guschinsky 2017b) integrated the machine configuration problem with process planning for the parts to be machined. The problem was still formulated as a MIP, with the objective of obtaining a desired cycle time at minimum cost. Similar MIP models and solution approaches have been proposed for transfer line balancing and optimisation by Dolgui et al. (2006, 2008) and Battaïa, Dolgui, and Guschinsky (2020). Afrin et al. (2016) extended the optimal design problem of rotary machines to multi-objective optimisation by simultaneously minimising the costs and the greenhouse gas emissions; they presented a meta-heuristic algorithm based on scatter and Tabu search procedure.

Huang, Süer, and Urs (2012) addressed the problem of scheduling a rotary injection moulding machine applied to the production of pairs of shoes. Their problem is similar to the one we consider, as the setup time required by the change of moulds and the mould availability are both taken into account; however, they do not have a limit on the maximum number of replications that a mould can sustain. In addition, while in our case the duration of each step in a sequence can vary and it is determined by the longest among the operations that are performed at the stations, in their case the duration of sequence steps is constant and it equals the longest injection time of the jobs on the machine. The authors proposed a sequential genetic algorithm and a heuristic procedure based on the differences in injection times.

2. Scheduling of Semi-Automatic carousel: an exponential-size model

We propose a MILP model where variables are associated with fixed *sequences* of s products, and they define the number of times each sequence is replicated within a so-called production *phase*. Once the RTM has completed the number of replications for a sequence within a specific phase, the machine is stopped and a new sequence is set up. If a sequence is replicated a large number of times, it may be necessary to stop the RTM, refurbish the moulds and start a new production phase. This is modeled as a further setup. In the model we propose, the setup time includes the real setup of the RTM, and the processing time of the first s steps after the machine is restarted, when no part is obtained (the first part is available at the end of the extraction phase, that is, at step $s+1$).

Let us denote by p a feasible sequence of s products, where products possibly appear more than once. A sequence is feasible if it is possible to define processing times for the s steps satisfying the technological constraints. In addition, the number of times a product appears within a sequence cannot exceed the number of available moulds. Let $\mathcal{P}$ be the set of all feasible sequences. For each $p \in \mathcal{P}$, we define an integer variable λ_p denoting the number of times sequence p is processed on the RTM; we also define a cost t_p, which is simply the time for processing sequence p once, when the RTM is setup and full (i.e., a first complete rotation has been performed). In addition, we also define an integer variable y_p representing the number of production phases for sequence p. This variable also defines the number of setups for p, each one having a setup cost (time) W_p. Let a_p^j, $p \in \mathcal{P}$, $j = 1, \ldots, N$, denote the number of parts of product j that are obtained by processing sequence p once. A MILP model reads:

Table 4. Problem parameters and variables.

Parameter	
s	number of stations in the carousel
N	number of different products
τ_j^p	preparation time for product j
τ_j^c	casting time for product j
τ_j^e	extraction time for product j
τ_j^{sl}	minimum solidification time of product j
τ_j^{su}	maximum solidification time of product j
τ_j^W	setup time for product j
U_j	demand of product j
Q_j	moulds availability for product j
$\mathcal{P}$	set of all feasible sequences
t_p	processing time of sequence p
W_p	setup time of sequence p
a_p^j	number of parts of product j that are obtained by processing sequence p once
M	maximum number of replications for a sequence before the moulds are refurbished
μ	minimum number of replications for a sequence
Variable	
λ_p	number of times sequence p is processed
y_p	number of setups for sequence p
y_{pi}	binary variable taking value 1 if sequence p is processed during the i-th production phase
d_t	duration of the generic t-th step of the sequence
x_{jt}	binary variables taking value 1 if the j-th product is launched at the t-th step

$$
\begin{aligned}
(\mathrm{ILP_E}) \qquad & \min \sum_{p\in\mathcal{P}} t_p\lambda_p + \sum_{p\in\mathcal{P}} W_p y_p && && (2)\\
& \sum_{p\in\mathcal{P}} a_p^j \lambda_p \geq U_j && j = 1,\dots,N, && (3)\\
& \lambda_p \leq M y_p && p \in \mathcal{P}, && (4)\\
& \lambda_p \geq \mu y_p && p \in \mathcal{P}, && (5)\\
& \lambda_p \in \mathcal{Z}^+ && p \in \mathcal{P}, && (6)\\
& y_p \in \mathcal{Z}^+ && p \in \mathcal{P}. && (7)
\end{aligned}
$$

The objective function (2) minimises the sum of processing time and setup time. Constraints (3) impose to satisfy the demand of each product. Each one of constraints (4) defines the number of setups y_p for the associated sequence p, where M in (4) is the maximum number of replications of a sequence that can be performed before the RTM is stopped and the moulds are refurbished. Constraints (5) instead impose that, after each setup, a sequence p is replicated at least μ times. Finally, (6) and (7) define the number of replications and the number of setups of each sequence as non-negative integers. If value M is large enough to allow each sequence to be processed during a single production phase, model ILP_E is equivalent to a cutting-stock formulation with setups and minimum production quantities for processed patterns.

In addition, in our formulation, the sequences implicitly encode a processing order for the products they include, which determines the sequence cost. For each selection of s products there can be up-to $(s-1)!$ sequences (permutations of the processing order); since products can be repeated in a sequence, the overall number of variables is upper bounded by the number of permutations with repetition of s out of N elements, that is, $|\mathcal{P}| = O(N^s)$. Nevertheless, it is enough to consider one sequence for each selection of s products, corresponding to the optimal permutation of the processing order. Hence, we can restrict to $O\binom{N+s-1}{s}$ variables. In addition, remember that feasible sequences include, for a given product j, no more than the number of available moulds Q_j.

Even if model (2)–(7) does not contain infeasible columns, it does not consider the availability of moulds in consecutive production phases, and hence it can happen that a solution cannot be feasibly scheduled on the RTM. Indeed, exhausted moulds need to be refurbished before re-use. This operation is time consuming and it is performed while the RTM is operating a production phase with a different set of moulds. This means that the overall number of moulds for a given product j that are used in two subsequent production phases of the RTM cannot exceed the number of moulds Q_j that are available for the product. In order to make sure a solution can be feasibly scheduled, we can introduce variables encoding the processing order of the sequences. Let y_{pi}, $p \in \mathcal{P}$, $i = 1,\dots,m$ be a binary variable taking value 1 if sequence p is processed during the i-th production phase, where m is an upper bound on the number of production phases, that we set to $\sum_{j=1,\dots,N} U_j/(\mu\, s)$. Model (2)–(7) is promptly extended in order to take into account the scheduling of production phases, as follows

$$\sum_{i=1}^{m} y_{pi} = y_p \qquad p \in \mathcal{P} \tag{8}$$

$$\sum_{p\in\mathcal{P}} a_p^j y_{p(i-1)} + \sum_{p\in\mathcal{P}} a_p^j y_{pi} \leq Q_j \qquad j = 1,\ldots,N,\ i = 2,\ldots,m \tag{9}$$

$$\sum_{p\in\mathcal{P}} y_{pi} \leq \sum_{p\in\mathcal{P}} y_{p(i-1)} \qquad i = 2,\ldots,m \tag{10}$$

$$\sum_{p\in\mathcal{P}} y_{pi} \leq 1 \qquad i = 1,\ldots,m \tag{11}$$

$$y_{pi} \in \{0,1\} \qquad i = 1,\ldots,m;\ p \in \mathcal{P}. \tag{12}$$

Equality (8) links the newly introduced y_{pi} variables with the y_p variables. Constraints (9) impose that no more than the available moulds are used in two consecutive production phases. Constraints (10) impose that a production phase can be activated only if the previous phase is activated, while constraints (11) impose that no more than one sequence is assigned to a production phase. Finally, (12) define the new variables as binary.

Before discussing the enumeration of the optimal sequences, associated with the variables to be considered in our formulation, we conclude this section by briefly discussing an alternative *descriptive* MILP model for the Scheduling of a Semi-Automatic carousel. It is clearly possible to define a polynomial number of binary variables assigning explicitly each product to its position in the production sequence (this is indeed the path we follow in the next section, where we define the structure of a sequence). However, this approach has three major drawbacks: i) it encodes a solution by means of a huge number of nonzero variables, because it defines which product is launched at each rotation of the RTM, and the number of such rotations is very large in a practical application; ii) it requires a quite involved modeling of set-ups and iii) it has a very weak linear relaxation, making its solution hard even for small instances, as we observed in some preliminary computational experiments we performed.

2.1. *Enumeration of optimal sequences (model variables generation)*

Although in exponential number, variables can be fully enumerated in very short time for instances of the problem having industrial relevance. For defining a sequence $p \in \mathcal{P}$ (and compute the cost t_p of the associated variable), we have to define:

i) the products in the sequence, that is, parameter $a_p^j,\ j = 1,\ldots,N$;
ii) the processing order within the sequence;
iii) the duration of each step in the sequence, which defines its cycle time.

In a `first` strategy, we enumerate all the combinations with repetitions of s products (i.e., we define the values of the parameter $a_p^j,\ j = 1,\ldots,N$) that do not exceed the availability of moulds, i.e., such that $a_p^j \leq Q_j,\ j = 1,\ldots,N$. Then, for each combination of products, we define the optimal permutation (i.e., processing order) and the duration of each step in the production cycle, by solving the following optimisation problem. The problem is strongly $NP-$hard, as discussed at the end of this section.

Problem statement: optimal permutation and step duration. Given a combination of s products (possibly with repeated products), we want to define the processing order within the sequence and the duration of each step in the sequence, in order to minimise the cycle time. We denote this problem as *permutation problem* in the reminder of this section.

The permutation problem can be modeled as the MILP that follows, where we optimise a sequence, i.e., a complete rotation of the RTM, when the machine is fully loaded. Let $t = 1, \dots, s$ be an index denoting the rotation step of the machine. The index is cyclic with period s, that is, $t - s = t = t + s$. The use of this index allows us to express conditions for $t < 1$, which are well defined by replacing it with $t + s$, and for $t > s$, well defined by replacing it with $t - s$. The model includes continuous variables $d_t,\ t = 1, \dots, s$, defining the duration of the generic t-th step of the sequence; and binary variables $x_{jt},\ j = 1, \dots, N;\ t = 1, \dots, s$ taking value 1 if the j-th product is launched (i.e., prepared at station 1) at the t-th step, and 0 otherwise. The MILP model for the permutation problem reads

$$\min \sum_{t=1}^{s} d_t \tag{13}$$

$$\sum_{j=1}^{N} x_{jt} = 1 \qquad t = 1, \dots, s \tag{14}$$

$$\sum_{t=1}^{s} x_{jt} = a_p^j \qquad j = 1, \dots, N \tag{15}$$

$$\tau_j^p x_{jt} \le d_t \quad j = 1, \dots, N,\ t = 1, \dots, s \tag{16}$$

$$\tau_j^c x_{j,t-1} \le d_t \quad j = 1, \dots, N,\ t = 1, \dots, s \tag{17}$$

$$\tau_j^e x_{j,t+1} \le d_t \quad j = 1, \dots, N,\ t = 1, \dots, s \tag{18}$$

$$\sum_{j=1}^{N} x_{jt}(\tau_j^{sl} + \tau_j^c + \tau_j^e) \le \sum_{k=1}^{s-1} d_{t+k} \qquad t = 1, \dots, s \tag{19}$$

$$\sum_{j=1}^{N} x_{jt}\tau_j^{su} \ge \sum_{k=2}^{s-2} d_{t+k} \qquad t = 1, \dots, s \tag{20}$$

$$\sum_{t=1}^{s} x_{jt} \le Q_j \qquad j = 1, \dots, N \tag{21}$$

$$x_{jt} \in \{0, 1\} \quad j = 1, \dots, N,\ t = 1, \dots, s. \tag{22}$$

Where the generic rotation step index l of the machine in variables x_{jl}, d_l is replaced as:

$$l = \begin{cases} l + s & \text{if } l < 1 \\ l - s & \text{if } l > s \end{cases}$$

The objective function (13) minimises the processing time of a full sequence of s steps. Constraints (14) impose that at each step only one product is prepared in station 1. Constraints (15) impose that the quantity of parts realised for each product

j by the sequence p is exactly the requested amount a_p^j. Clearly, the model has a feasible solution if and only if $\sum_{j=1}^{N} a_p^j \leq s$. Stations 1, 2 and s are dedicated to the preparation, the casting and the extraction of the parts, respectively; therefore, the duration of the time step has to allow the completion of such operations, as expressed in constraints (16), (17) and (18), respectively. As previously explained, a part solidifies and cools down in all the stations between 3 and $s-1$. Furthermore, the time spent in station 2 after pouring and in station s before the extraction contributes to the solidification of the part. The whole time of the part in the mould must be at least the minimum solidification time τ_j^{sl}; this condition is imposed by constraint (19). As well, the time of the part within the mould must not exceed the upper limit τ_j^{su}. As the casting and extraction can be performed so that no cooling happens in stations 1 and s, these stations can be omitted in the formulation of the condition, that can be written as in constraint (20).

A `second` strategy for defining all the sequences is depicted in Algorithm 1. We enumerate all the combinations with repetitions of s products that do not exceed the availability of moulds, as in the `first` strategy. For each combination, we consider all permutations of the order in which the products are processed on the RTM. Given a permutation σ, it remains to compute the optimal values of the d_t, $t = 1, \ldots, s$ variables, and the cost of the permutation. This problem can be formulated as a linear program, which is equivalent to (13)–(22) once the value of the x_{jt} variables is fixed by the considered permutation, and hence it has polynomial time complexity.

For each combination of products, we define a variable whose cost is the cost of the best permutation (if one exists satisfying the cycle time constraints).

Algorithm 1 Column Enumeration

```
1: for each selection of products satisfying the number of available moulds do
2:     set $t_p = +\infty$
3:     for each permutation $\sigma = \sigma(1), \ldots, \sigma(s)$ of the products order do
4:         set $t_\sigma = \sum_{l=1}^{s} d_l = \sum_{l=1}^{s} \max\{\tau^p_{\sigma(l)}, \tau^c_{\sigma(l-1)}, \tau^e_{\sigma(l+1)}\}$ (with $\sigma(l+s) = \sigma(l)$);
5:         if $t_\sigma \geq t_p$ then
6:             discard $\sigma$ and continue with next permutation;
7:         if $\sigma$ is such that the solidification time exceeds $\tau_j^{su}$ for some $j$ then
8:             discard $\sigma$ and continue with next permutation;
9:         if $\sigma$ is such that the solidification time is smaller than $\tau_j^{sl}$ for some $j$ then
10:            recompute $t_\sigma$ heuristically or by solving (13)–(22) with $x_{jt}$ fixed by $\sigma$;
11:        if $t_\sigma < t_p$ then
12:            $t_p = t_\sigma$;
13:        if $t_p = LB$ then
14:            break;
15:    end for
16:    if $t_p < +\infty$ then
17:        store sequence $p$ with associated variable $\lambda_p$ of cost $t_p$
18: end for
```

Algorithm 1 describes the enumeration of the problem variables (columns). We assume to start with a fully loaded RTM. We perform a complete rotation (s steps) with the permutation σ at study, and we define, in line 4, the duration d_l of each step

$l = 1, \dots, s$ as the maximum among the preparation, pouring and extraction times. In the expression, the product in position $\sigma(l)$ is being prepared, the one in position $\sigma(l-1)$ is being poured, and the one in position $\sigma(l+1)$ is being extracted and will be prepared at the next step; if the index is below 1 or exceeds s, its right value is obtained by reminding that the permutation repeats identical every s steps. This way, the step duration d_l satisfies constraints (16), (17) and (18). We also obtain the cost of the permutation as sum of the steps duration; the permutation is immediately discarded if not improving on the best known permutation (line 6). If the defined step durations are such that the maximum admissible casting solidification time is exceeded for any product (constraint (20)), the permutation is discarded (line 8). If the minimum admissible duration for casting solidification is satisfied for each product (constraint (19)), the cost of the permutation is confirmed. Otherwise, the optimal duration d_t of each step can be obtained by solving the linear program (13)–(22) after fixing the value of the x_{jt} variables (line 10). As an alternative to the solution of the linear program, one can heuristically increase the duration of the steps. If the overall time increase for the processing time of the sequence equals the largest increase in the cooling time needed by one product in the sequence, the heuristic increase is optimal. Finally, if a valid lower bound LB on the cost of the best permutation is known, the procedure stops as soon as this value has been found (line 13). The theoretical run time of the algorithm is reported in the following observation.

Observation 2.1. In the worst case, Algorithm 1 asks to solve $O(N^s)$ linear programs equivalent to (13)–(22) with fixed x variables. For fixed s, Algorithm 1 has a polynomial run time.

Next, we discuss possible lower bounds on the cost of the best permutation for a given combination of products.

Proposition 2.2. *A valid lower bound on the processing time of a combination including products $j = 1, \dots, s$ with processing parameters τ_j^p, τ_j^c, τ_j^e, is*

$$LB_1 = \max\{\sum_{j=1}^{s} \tau_j^p; \sum_{j=1}^{s} \tau_j^c; \sum_{j=1}^{s} \tau_j^e\} \tag{23}$$

Proposition 2.3. *Let consider a combination of s products with processing parameters as in Proposition 2.2. Let $\sigma(.)$ be a permutation of the product indices defining a non-increasing ordering of parameter τ_j^p. Similarly, define permutations $\rho(.)$ and $\psi(.)$ for parameters τ_j^c and τ_j^e, respectively. A valid lower bound on the processing time of the combination is*

$$LB_2 = \sum_{j=1}^{s} \max\{\tau_{\sigma(j)}^p; \tau_{\rho(j)}^c; \tau_{\psi(j)}^e\} \tag{24}$$

Proof. Processing a sequence requires s steps. The duration of a step t is determined by the largest among τ_j^p, τ_i^c and τ_l^e where j is the product at preparation, i at pouring and l at extraction. LB_2 is the cost of the ideal situation in which the longest preparation is executed together with the longest pouring and the longest extraction, the second longest preparation is executed together with the second longest pouring and the second longest extraction, etc. □

Proposition 2.4.

$$LB_1 \leq LB_2 \tag{25}$$

Proof. Since permuting the order of the addends of a summation does not change the result, we have $LB_1 = \max\{\sum_{j=1}^{s} \tau_j^p; \sum_{j=1}^{s} \tau_j^c; \sum_{j=1}^{s} \tau_j^e\} = \max\{\sum_{j=1}^{s} \tau_{\sigma(j)}^p; \sum_{j=1}^{s} \tau_{\rho(j)}^c; \sum_{j=1}^{s} \tau_{\psi(j)}^e\} \leq \sum_{j=1}^{s} \max\{\tau_{\sigma(j)}^p; \tau_{\rho(j)}^c; \tau_{\psi(j)}^e\} = LB_2$, where the inequality holds by definition of maximum.

□

After enumeration, columns that due to limited mould availability are not compatible with any other column can be removed from $\mathcal{P}$.

2.2. *Complexity*

The complexity of the permutation problem and of the overall scheduling of the semi-automatic carousel are discussed in the next propositions.

Proposition 2.5. *Given s products to be included in a sequence (where s is part of the input), the permutation problem, that is, the problem of defining the optimal permutation and duration of each step, is strongly $NP-$hard.*

Proof. The result follows from reduction from the strongly $NP-$hard *three-machine synchronous flow shop with makespan minimization* ($F3|synmv|Cmax$) (see Waldherr and Knust (2015)). In this problem we are given three machines and a set of jobs J, each job j consisting of three operations to be performed in sequence on the machines, with times (p_{1j}, p_{2j}, p_{3j}). The transfer of jobs between the machines is synchronous and takes place when all machines have completed the current operation. The decision version of the problem asks if there is a sequencing σ of the jobs with a makespan at most M. We show that $F3|synmv|Cmax$ reduces to an instance of the permutation problem with $|J| + 3$ products and stations, where $\tau_j^{sl} = 0$ and $\tau_j^{su} = +\infty$ for each product $j = 1, \ldots, |J| + 3$. In the reduction, jobs correspond to products and machines to positions on the rotary machine. In addition to the $|J|$ products (jobs), each one having its original processing time on the first three positions (machines), we define three dummy products with processing times $(0, 0, Q)$, $(0, Q, 0)$ and $(Q, 0, 0)$ for the first three positions, where $Q > M$. The processing times of all products are null for positions $4, \ldots, |J| + 3$.

We show that the $F3|synmv|Cmax$ has makespan at most M if and only if there exists a permutation on the rotary machine with cycle time at most $Q + M$. Let σ be a sequence for which $F3|synmv|Cmax$ has makespan at most M. Append the three dummy products at the end of the sequence in the order they are given above, thus obtaining a sequence σ'. Sequence σ' can be processed on the rotary machine with a cycle time at most $Q + M$, because the dummy products only need an additional Q processing time when processed in the proper order. Viceversa, assume a sequence σ' for the rotary machine has processing time at most $Q + M$. The three dummy products must be processed in their order (this minimizes idle time) and define a step of length Q when they are in the first three positions of the rotary machine, which leaves a time at most M to process the original products on the first three positions (while all other positions require null time). □

Table 5. RTM configuration during the 6 steps of sequence $\sigma' = [A, B, C, D^1, D^2, D^3]$.

position	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6
1	$\boldsymbol{A}$	$\boldsymbol{B}$	$\boldsymbol{C}$	D^1	D^2	$\boldsymbol{D^3}$
2	D^3	$\boldsymbol{A}$	$\boldsymbol{B}$	$\boldsymbol{C}$	D^1	$\boldsymbol{D^2}$
3	D^2	D^3	$\boldsymbol{A}$	$\boldsymbol{B}$	$\boldsymbol{C}$	$\boldsymbol{D^1}$
4	D^1	D^2	D^3	A	B	C
5	C	D^1	D^2	D^3	A	B
6	B	C	D^1	D^2	D^3	A
d_t	p_{1A}	$\max\{p_{2A}, p_{1B}\}$	$\max\{p_{3A}, p_{2B}, p_{1C}\}$	$\max\{p_{3B}, p_{2C}\}$	p_{3C}	Q

In Table 5 we report a numerical example to illustrate the proof of Proposition 2.5. The $F3|synmv|Cmax$ has three jobs, whose permutation $\sigma = [A, B, C]$ can be processed with makespan M on the machines (positions) $1, 2, 3$. By appending the dummy jobs D^1, D^2, D^3 we obtain sequence $\sigma' = [A, B, C, D^1, D^2, D^3]$ which has cycle time on the RTM not larger than $M + Q$. For each step, we indicate in bold the product(s) (jobs) determining the duration of the corresponding step.

Proposition 2.6. *The overall scheduling of a Semi-Automatic carousel is strongly $NP-hard$.*

Proof. The permutation problem is a special case of the overall optimization of the Semi-Automatic carousel production which occurs when i) a single sequence is sufficient to obtain the whole production and ii) setup costs are large, hence no set-up takes place. □

3. Heuristic algorithm

In this section we discuss a heuristic approach for solving formulation ILP_E. We assume the set (or a subset) of variables $\mathcal{P}$ is given. The idea is to incrementally construct a feasible solution by selecting, at each iteration:

i) the best sequence $p \in \mathcal{P}$ and
ii) the best increase Δ_{λ_p} to the current value of variable λ_p, according to a specified score $\sigma(p, \Delta_{\lambda_p})$;
iii) a feasible position in the production phases schedule (w.r.t. moulds availability) for the new phase, if a new setup is required to perform the increase Δ_{λ_p}.

A feasible solution is denoted as a *cover*, because each row j in constraints (3) of ILP_E must be *covered* at least U_j times by the products in the selected columns. Each column p, when selected, covers row j for a_p^j times. Initially, we have $\lambda_p = 0,\ p \in \mathcal{P}$, while during the iterations of the algorithm we have a partial cover with $\lambda_p, y_p > 0$ for some $p \in \mathcal{P}$. Each produced sequence p with $\lambda_p > 0$ has also one production phase position (or more than one if $\lambda_p > M$) within the production schedule. When a new setup is needed to perform an increase Δ_{λ_p} for some $p \in \mathcal{P}$, then a feasible *position* where to insert the new production phase, if any, is determined by Algorithm 2. In the algorithm, we denote two production phases as *incompatible* if they exceed the availability of moulds for some item when considered together. The algorithm returns a feasible position where to insert the production phase (by scaling the other phases) if one exists, and a negative value otherwise.

Algorithm 2 Find *position* for a production phase

```
1:  Let $l$ be the number of phases in a partial cover;
2:  set $position := l + 1$;
3:  repeat
4:      if the phase is incompatible with the one in $position - 1$ then
5:          $position := position - 2$;
6:      else if the phase is incompatible with the one in $position + 1$ then
7:          $position := position - 1$;
8:      else
9:          return $position$;
10: until $position \leq 0$
11: return $position$.
```

Observation 3.1. Algorithm 2 verifies that no feasible position exists for some sequence $p \in \mathcal{P}$ in at least $\lfloor l/2 + 1 \rfloor$ and at most l iterations.

Given a current solution (λ, y), a possible score for evaluating an increase Δ_{λ_p} of variables λ_p, is

$$\phi(p, \Delta_{\lambda_p}) = \frac{\sum_{j=1,\ldots,N} \min[\tilde{U}_j, \Delta_{\lambda_p} a_p^j]}{\Delta_{\lambda_p} t_p + \Delta_{y_p} W \alpha} \tag{26}$$

where $\tilde{U}_j = \max\{0, U_j - \sum_{j=1,\ldots,N} \lambda_p a_p^j\}$ is the residual demand of product j with respect to the current solution, Δ_{y_p} is the increase in number of setups and $\alpha > 0$ is an input parameter that calibrates the weight given to the two involved terms at the denominator (i.e., processing time and setup time). Clearly, for a given $p \in \mathcal{P}$, we do not consider all integer values $\Delta_{\lambda_p} = 1, \ldots, \lceil \max_{j=1,\ldots,N} \tilde{U}_j / a_p^j \rceil - \lambda_p$, but we restrict to those values of Δ_{λ_p} in the interval that either:

- would saturate the demand of a product in p;
- or would deplete the moulds in sequence p and then a setup would be needed for any additional production with the same sequence;

while satisfying the constraint on the minimum number of sequence replication for each production phase if a new production phase needs to be activated.

3.1. *Intensification*

Each time a feasible solution is found, an intensification step is run in order to possibly reduce its cost. For each column $p \in \mathcal{P}$ and for each production phase of the current feasible solution, associated with a sequence p', we evaluate an alternative solution: we remove the production phase (and set $\lambda_{p'} = \lambda_{p'} - \min\{\lambda_{p'} - \mu\,(y_p - 1), M\}$) and replace it with a production phase built with column p. This operation is possible only if column p contains a sequence of moulds that can satisfy the residual demand for each product, and if the position of the production phase in the schedule is feasible for column p (w.r.t. moulds availability). Variable λ_p is set at the minimum value for covering the residual demand. At each iteration, we perform the best switch among all the feasible ones (*best-first search*) and we stop when no improving switch is possible.

By means of this intensification step, we also guarantee that the cover we found is minimal, i.e., there is no variable λ_p, $p \in P$ whose value can be reduced without violating a demand constraint.

3.2. *Metaheuristic diversification*

As a single run of the described heuristic is very fast, we use it within a classical *iterated local search* scheme in order to diversify the search. More precisely, as described in Algorithm 3, each local search iteration is based on the *ruin and recreate* paradigm. The recreate step corresponds to the heuristic algorithm described in section 3, which is run in a *GRASP* fashion (Feo and Resende 1995): all pairs (p, Δ_{λ_p}), $p \in \mathcal{P}$ are ranked according to the non-increasing value of their score function (26). Given a parameter $\beta \in [0, 1]$, a random pair $(p^*, \Delta_{\lambda_{p^*}})$ among the fraction of the first β pairs in the ranking is then selected, and the value of variable λ_{p^*} is increased by $\Delta_{\lambda_{p^*}}$. This operation is repeated until a feasible solution is built (i.e., until all the demands are satisfied). After the intensification step, the incumbent is possibly updated. Then, the ruin step destroys part of the solution: all the production phases are ordered according to the total number of parts that exceed some product demand, and, starting from the most overproducing phase, the procedure destroys the first one that can be removed, without violating the constraints on the number of available moulds in consecutive phases. The ruin operation is iterated until the resulting partial cover does not cover any product more than needed or at least half of the production phases have been destroyed. A single local search loop terminates when the solution value does not improve after a ruin-and-recreate iteration. The iterated local search algorithm terminates when a stopping criterion is met (time limit and/or a maximum number of local search iterations).

Observation 3.2. The number of iterations of Algorithm 3 depends on the imposed stopping criteria. The time complexity of one iteration of its most time-consuming operation, namely, the *recreate* phase, is given by $O(s^2|\mathcal{P}| + |\mathcal{P}| \log |\mathcal{P}|)$, since at most s quantity increases are evaluated for each $p \in \mathcal{P}$, each one having cost $O(s)$, and a sorting operation of the $|\mathcal{P}|$ variables is performed.

4. Computational experiments

In this section, we report on computational experiments performed on a set of realistic instances that represents a typical industrial scenario. We generated our instances starting from real technological parameters and production demands provided by an aluminum foundry using a RTM for production within a Aluminum GDC process. We implemented a column generation procedure based on Algorithm 1, this way we make available all the feasible variables of the problem. Through our experiments we evaluate the performances of the described solution methods; in addition, we analyse the obtained solution structure, which allows us to validate the problem formulation presented in Section 2.

Algorithm 3 Metaheuristic algorithm(α, β, *stopping_criteria*)

```
1: repeat                                                          ▷ iterated greedy loop
2:     set $\lambda_p = 0$ for each $p \in \mathcal{P}$;
3:     repeat                                                      ▷ local search loop
4:         $\mathcal{Q} = \emptyset$;
5:         repeat                                                  ▷ recreate
6:             find the best score $\sigma^*(q, \Delta_{\lambda_q})$ for each $q \in \mathcal{P}$ for which a feasible
7:             phase position to perform $\Delta_{\lambda_q}$ exists; $\mathcal{Q} = \mathcal{Q} \cup q$;
8:             sort elements of $\mathcal{Q}$ according to decreasing values of $\sigma^*(q, \Delta_{\lambda_q}, \alpha)$;
9:             randomly select $q^*$ from the first $\beta\,|\mathcal{Q}|$ elements of $\mathcal{Q}$;        ▷ GRASP
10:            set $\lambda_{q^*} := \lambda_{q^*} + \Delta_{\lambda_{q^*}}$;
11:        until $\sum_{p \in \mathcal{P}} a_p^j \lambda_p \geq U_j \, \forall\, j \in N$
12:        repeat                                                  ▷ intensification
13:            perform the best phase switch;
14:        until no improving switch is possible
15:        repeat                                                  ▷ destroy
16:            randomly remove a phase.
17:        until ( $\sum_{p \in \mathcal{P}} a_p^j \lambda_p \leq U_j \, \forall\, j \in N$ or half of the phases have been destroyed)
18:    until solution value does not improve
19: until stopping_criteria
```

All the reported experiments were performed on a computer equipped with a AMD 3960X processor clocked at 3.8 GHz and 128 GB RAM under Linux operating system. We used the Gurobi 9.1 MIP framework to solve the problem formulations for which we report the results. Gurobi was run in single-threaded mode and all Gurobi parameters were set to their default values.

4.1. *Benchmark instances.*

Starting from real data from an aluminum foundry, we evaluate the minimum and maximum value for each component of the τ vector of the characteristic cycle times (see (1)); from the same data, we evaluate the minimum and maximum value of the mould setup time τ^W, of moulds availability Q, and of the product demand U associated with a production period. These values are reported in Table 6, all the values but those of Q and U are expressed in seconds. From these data we then obtain realistic instances, where for each product j the actual values of each entry of vector τ_j, τ_j^W, Q_j and U_j are integer numbers randomly generated within the range $[\min, \max]$. Product demands are multiples of 100. We recall that setup times W_p are sequence dependent: for each sequence we set $W_p = \max_{j \in p} \tau_j^W + t_p$, where we add to the "real" setup time the additional term t_p, since no part is extracted during the first s steps of rotation.

Table 6. Ranges from which product-dependent parameters are randomly generated.

	τ^p	τ^c	τ^e	τ^{sl}	τ^{su}	τ^W	Q	U
min	45	20	40	160	220	3600	1	300
max	90	40	60	180	240	4800	4	1500

We consider instances with a number of products $N \in \{10, 11, 12, 13, 14, 15\}$; for the number of station in the RTM, we consider $s \in \{5, 6\}$, in two variants: either all stations must be loaded in a sequence, or one of the stations can be left empty. By generating 10 instances for each configuration, we obtain 240 instances in total.

4.2. *Results and Discussion*

In this section we initially consider results obtained by solving model (2) – (12) with the Gurobi MILP solver. This approach will be referred as `GRB` in the following.

Table 7 collects the results obtained by `GRB` on the benchmark instances when solving the problem by considering either a *fully loaded* RTM (top part of the table) or by allowing up to one *empty station* on the RTM (bottom part of the table). Indeed, the characteristic cycle times (1) are such that a RTM with 5 or 6 stations achieves a good balance, but alternating between 5 and 6 stations requires a long reconfiguration time and it is impractical. Then in the industrial practice, when a 6 station configuration is chosen, it happens that sometimes one (or more) stations are left empty, in order to reduce overproduction. The *empty station* configuration is obtained by adding a dummy product having no demand (i.e., $U_j = 0$), an infinite mould availability (i.e., $Q_j = \infty$) and processing times with no effect on the production process (i.e., $\tau_j^p = 0$, $\tau_j^c = 0$, $\tau_j^e = 0$, $\tau_j^{sl} = 0$, $\tau_j^{su} = \infty$ and $\tau_j^W = 0$). We considered a time limit of 1 hour, each line is relative to 10 instances with the same features. Columns "N" and "s" of the table report the number of products and stations, respectively; column "#part" reports the average demand calculated over the 10 instances; column "#var" reports the average number of generated columns; column "#cons" reports the average number of constraints. Then, the rest of the columns provide information on the solution process and on the solution structure: columns "gen[s]", "gap", "opt" are performance related information and they report the time (in seconds) required to generate the columns, the optimality gap of Gurobi at time limit (computed as $(UB - LB)/LB \times 100$, where UB is the cost of the best feasible solution found and LB is the lower bound on the optimal solution cost at time limit), and the number of instances solved to optimality, respectively. Columns "Cmax[h]", "%setup", "#setup" and "%over" are information related to the solution structure and they report the best known solution cost (makespan in hours); the percentage of the time spent to set up the RTM (calculated as (setup time)/(total time)$\times$100), the number of performed setups and the percentage of overproduction (calculated as (total production - total demand)/total production $\times$100), respectively. All the reported values (but those of column "opt") are average values over 10 considered instances. For both the cases (i.e., *fully loaded* and *empty station*) there is an additional row providing the total average values.

The results of Table 7 show how the complexity of the problem quickly increases with the size of the considered instance, even if the time required to generate all the variables remains almost negligible (the highest value being equal to 0.32 seconds, while the time needed to write the MILP model is of the order of milliseconds). Indeed, as already pointed out in Section 2, the number of generated variables exponentially grows with respect to N and s; this can be observed from column "#var" whose values start from few thousands in the smaller instances, and reach tens of thousands in larger ones. Therefore, even a small increase of N or s translates into much more challenging instances: 15 out of the 16 instances solved to optimality (column "#opt") belong to groups of instances with at most 11 products (the average time for solving these instances is equal to 1302.1 seconds in the "fully loaded" case and 1154.01 seconds in the "empty station" case). Similarly, the percentage optimality gap is always larger than 2 when $N \in \{12, 13, 14, 15\}$ and $s = 6$.

Our experiments give some useful insight of the use of empty stations on the RTM. Clearly this is a further optimisation opportunity, and hence the cost of the optimal solution for the problem when empty stations are allowed is smaller than or equal to the cost of the optimal solution when stations are full. However, column "best[h]" reveals that allowing an empty station on the RTM and, consequently, addressing a more difficult instance does not help in saving processing time: both the configurations lead to feasible solutions with an average objective value slightly larger than 224 hours; for both the cases, little more than 7% is made up of setup times. Since these figures correspond to solutions with very small optimality gap, we do not expect the use of empty stations to be advantageous even if one was able to solve the problem to optimality. Allowing an empty station seems to be useful only if one aims to reduce the undemanded production. Indeed, with this configuration the value of "%over" is reduced by more than half (from 0.96 to 0.42).

Since the structure of the solutions obtained by the two different configurations is similar, we simply consider fully loaded stations when comparing `GRB` with the metaheuristic algorithm described in Section 3.2. In order to push to the limit the two solution approaches, we also enlarge our test-bed by considering additional instances with $N \in \{16, 17, 18, 19, 20\}$. This way, we have a total of 220 instances in this comparison. We will refer to the the metaheuristic algorithm as `HEUR` in the following. As `HEUR` only provides feasible solutions (with no lower bound), the comparison is

Table 7. Performances and solution structure results obtained by GRB over 240 considered instances.

	N	#part	s	#var	#cons	gen[s]	gap	opt	Cmax[h]	%setup	#setup	%over
fully loaded	10	9310	5	1301.1	2829.6	0.02	0.75	3	185.89	4.20	5.7	0.72
			6	2134.0	4459.4	0.03	0.76	3	186.23	3.96	5.4	3.02
	11	10290	5	2193.1	4658.8	0.03	1.10	1	206.51	4.21	6.5	0.30
			6	3911.4	8047.3	0.04	1.22	1	206.09	3.85	5.9	1.55
	12	9950	5	3363.7	7010.0	0.04	1.44	0	199.01	4.49	6.8	0.31
			6	6678.6	13595.0	0.07	2.02	1	199.78	4.21	6.3	2.02
	13	11250	5	4858.3	10058.6	0.06	1.77	0	221.67	4.42	7.3	0.71
			6	10732.3	21751.1	0.10	2.31	0	218.54	4.33	7.0	1.18
	14	12760	5	6546.7	13509.6	0.07	1.86	0	250.76	4.47	8.3	0.44
			6	15654.1	31654.0	0.15	2.37	0	249.73	4.49	8.3	0.46
	15	14190	5	9228.9	18946.1	0.11	1.92	0	283.20	4.50	9.7	0.30
			6	22664.2	45735.1	0.23	2.53	0	281.34	4.36	9.3	0.53
						0.08	1.67	9	224.06	4.29	7.2	0.96
empty station	10	9310	5	1850.5	3928.4	0.03	0.65	3	185.71	4.14	5.6	0.60
			6	3358.9	6909.2	0.03	1.05	2	186.15	4.06	5.6	1.00
	11	10290	5	3021.1	6314.8	0.04	0.96	1	206.33	4.08	6.3	0.31
			6	5953.0	12130.5	0.06	1.66	1	206.33	4.02	6.2	0.39
	12	9950	5	4538.7	9360.0	0.05	1.52	0	199.04	4.52	6.9	0.27
			6	9840.5	19918.8	0.09	2.47	0	200.04	4.51	6.8	0.35
	13	11250	5	6438.4	13218.8	0.08	1.86	0	221.88	4.73	7.8	0.42
			6	15365.5	31017.5	0.15	2.43	0	218.29	4.46	7.2	0.32
	14	12760	5	8571.2	17558.6	0.10	1.76	0	250.50	4.51	8.4	0.30
			6	21916.4	44178.6	0.22	2.66	0	250.23	4.60	8.5	0.48
	15	14190	5	11895.2	24278.7	0.14	1.80	0	282.87	4.37	9.4	0.25
			6	31430.9	63268.5	0.32	2.76	0	281.68	4.55	9.7	0.35
						0.11	1.80	7	224.09	4.38	7.4	0.42

performed on the quality of those solutions only, whose optimality gaps are computed with respect to the best lower bound LB obtained by Gurobi in our experiments. Preliminary experiments proved that the best setting of HEUR is obtained with $\alpha = 20$ and $\beta = 0.005$. Table 8 compares the average percentage gaps reached by GRB and HEUR over the 220 considered instances. As for Table 7, columns "N" and "s" report the number of products and stations, respectively; column "#part" reports the average demand computed over the 10 instances; column "#var" reports the average number of variables (sequences); column "#cons" reports the average number of constraints. Then, columns "1 min", "5 min" and "1 hour" report for both the methods the optimality gap of the best feasible solution returned by each method after 1 minute, 5 minutes and 1 hour of computing time, respectively. We highlighted in bold the values that are the best in the comparison. As a further improvement, we also implemented a hybrid approach (column "HYB") where we feed the solution obtained by HEUR after five minutes as a MIP start to Gurobi, and then we run Gurobi for 55 minutes; this way, a fair comparison with columns "1 hour" can be drawn. The last row of the table collects the total average values.

Table 8. Comparison of the average percentage gaps between GRB and HEUR over 120 considered instances.

					GRB			HEUR			HYB
N	#part	s	#var	#cons	1 min	5 min	1 hour	1 min	5 min	1 hour	5+55 min
10	9310	5	1301.1	2829.6	**1.62**	**0.96**	**0.75**	2.06	1.54	1.32	0.65
		6	2134.0	4459.4	**1.86**	**1.33**	**0.76**	2.97	1.95	1.71	0.85
11	10290	5	2193.1	4658.8	2.02	**1.56**	**1.10**	**1.81**	1.57	1.24	0.85
		6	3911.4	8047.3	**2.31**	**1.94**	**1.22**	3.18	2.37	1.75	1.17
12	9950	5	3363.7	7010.0	2.81	2.19	**1.44**	**2.27**	**1.94**	1.59	1.38
		6	6678.6	13595.0	3.52	2.44	2.02	**2.67**	**2.23**	**1.53**	1.85
13	11250	5	4858.3	10058.6	3.13	2.45	**1.77**	**2.71**	**2.33**	1.97	1.54
		6	10732.3	21751.1	6.13	**3.08**	2.31	**4.35**	3.32	**2.21**	2.17
14	12760	5	6546.7	13509.6	3.21	**2.44**	**1.86**	**3.20**	2.80	2.25	1.75
		6	15654.1	31654.0	39.96	3.24	2.37	**4.31**	**3.17**	**2.27**	2.18
15	14190	5	9228.9	18946.1	3.39	**2.47**	**1.92**	**2.88**	2.50	1.94	1.80
		6	22664.2	45735.1	78.71	3.43	2.53	**3.79**	**2.94**	**2.13**	2.62
16	13990	5	11985.7	24483.2	5.25	3.31	**2.32**	**3.42**	**2.97**	2.40	2.00
		6	28089.3	56604.0	91.74	44.79	3.27	**5.13**	**4.18**	**2.87**	2.88
17	15720	5	16755.2	34115.4	46.37	3.19	**2.42**	**3.76**	**3.19**	2.60	2.46
		6	44822.0	90148.3	83.75	83.75	2.95	**4.32**	**3.60**	**2.59**	2.87
18	15170	5	21544.1	43703.2	70.05	3.91	**2.94**	**4.33**	**3.85**	3.20	2.72
		6	63089.4	126691.8	78.44	78.44	3.75	**5.06**	**4.71**	**3.69**	3.14
19	17340	5	28420.8	57579.9	78.45	19.89	**2.65**	**4.88**	**4.26**	3.40	3.10
		6	90561.5	181739.5	82.82	68.95	51.06	**4.99**	**4.41**	**3.45**	3.88
20	18480	5	35020.9	70863.8	84.85	76.44	**2.88**	**4.22**	**3.61**	3.01	3.01
		6	109991.0	220669.8	90.39	90.39	57.85	**5.79**	**4.82**	**3.94**	4.13
					39.13	22.75	6.91	**3.73**	**3.10**	**2.41**	2.23

By comparing columns “1 min” and “5 min” for the first two approaches, we immediately observe that these computing times are too short for GRB, that seems able to provide good solutions only when the number of products is small (lower than 16); after this threshold, the solver struggles and produces solutions with an average percentage gap as large as 91.74 in the worst case. On the contrary, HEUR always finds solutions of acceptable quality, and it wins the comparison in 19 of out 22 cases with 1 minute of time limit, and in 15 out of 22 cases when the time limit is equal to 5 minutes. When a larger computing time is allowed (1 hour), the performances are more balanced. GRB wins the comparison in 13 out of 22 cases, but when we compare the total average percentage gaps, HEUR still appears as the best choice between these two methods. The results obtained by HYB highlight the benefits that can be obtained by combining the two approaches; indeed, it prevents Gurobi from having large optimality gaps and, in several cases (12 out of 22), it further reduces the average gap of the best between GRB and HEUR.

In summary, the experiments have shown that HEUR is capable of producing good quality solutions in short computing time, and to keep a good performance for large instances as well; instead, GRB struggles when the number of variables grows, and may need a long time to find feasible solutions of good quality, if it is not fed with a MIP start. In addition, an industrial user may appreciate a solution approach that does not require the use of external commercial software. These considerations make HEUR a useful tool for the optimization of RTM machines in an industrial Aluminium GDC setting.

5. Conclusions

Despite the relevance of rotary transfer machines in the aluminum gravity die casting industry, to the best of authors' knowledge the problem of optimally defining the cycle time and the production scheduling has not been previously investigated in the literature. In this paper we have started to fill this gap, by proposing a mixed-integer linear program for defining the optimal sequencing and scheduling a rotary transfer machines in the aluminum gravity die casting industry. In addition to a procedure for generating the variables of the formulation, we have also described a heuristic approach to find good quality solutions to the problem, without recurring to a general-purpose commercial solver. Extensive computational experiments on a set of instances generated from real data have assessed the computational effort needed to solve the aforementioned model, for which near optimal solutions of instances of industrial relevance can be computed within computing times that are totally compatible with a weekly scheduling. The experiments have also given insight on the optimal choice in the set-up of the stations on the rotary machine.

Future works may evaluate the possibility of extending the described approaches to similar problems. Indeed, addressing the enumeration of the product sequences and their scheduling on the machine as two independent problems, makes our solution strategy a flexible tool that may be naturally adapted to other industrial scenarios involving fixed production sequences. A further line of research could consider on-the-fly generation of the product sequences, to tackles those cases in which full enumeration is not a viable option.

Data availability statement

The data that support the findings of this study are available from the corresponding author, E.M., upon reasonable request.

Acknowledgements

Authors would like to thank the MIUR (Italian Ministry of University and Research) for funding support and Atti Foundry S.p.A for the collaboration during the development of the presented approach.

References

Afrin, Kahkashan, Ashif Sikandar Iquebal, Sri Krishna Kumar, M. K. Tiwari, Lyes Benyoucef, and Alexandre Dolgui. 2016. "Towards green automated production line with rotary transfer and turrets: a multi-objective approach using a binary scatter tabu search procedure." *International Journal of Computer Integrated Manufacturing* 29 (7): 768–785.

Battaïa, Olga, Alexandre Dolgui, and Nikolai Guschinsky. 2016. "Heuristics for Batch Machining at Reconfigurable Rotary Transfer Machines." *IFAC-PapersOnLine* 49 (12): 491–496.

Battaïa, Olga, Alexandre Dolgui, and Nikolai Guschinsky. 2020. "Optimal cost design of flow lines with reconfigurable machines for batch production." *International Journal of Production Research* 58 (10): 2937–2952.

Battaia, Olga, Alexandre Dolgui, Nikolai Guschinsky, and Genrikh Levin. 2012. "Optimal design of rotary transfer machines with turrets." *IFAC Proceedings Volumes* 45 (6): 407–412.

Battaïa, Olga, Alexandre Dolgui, Nikolai Guschinsky, and Genrikh Levin. 2013. "Parallel Machining of Multiple Parts on Rotary Transfer Machines with Turrets." *IFAC Proceedings Volumes* 46 (9): 1477–1482.

Battaïa, Olga, Alexandre Dolgui, Nikolay Guschinsky, and Genrikh Levin. 2012. "A decision support system for design of mass production machining lines composed of stations with rotary or mobile table." *Robotics and Computer-Integrated Manufacturing* 28 (6): 672–680.

Battaïa, Olga, Alexandre Dolgui, and Nikolai Guschinsky. 2017a. "Decision support for design of reconfigurable rotary machining systems for family part production." *International Journal of Production Research* 55 (5): 1368–1385.

Battaïa, Olga, Alexandre Dolgui, and Nikolai Guschinsky. 2017b. "Integrated process planning and system configuration for mixed-model machining on rotary transfer machine." *International Journal of Computer Integrated Manufacturing* 30 (9): 910–925.

Battaïa, Olga, Alexandre Dolgui, Nikolai Guschinsky, and Genrikh Levin. 2014. "Combinatorial techniques to optimally customize an automated production line with rotary transfer and turrets." *IIE Transactions* 46 (9): 867–879.

Belov, Gleb, and Guntram Scheithauer. 2006. "A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting." *European journal of operational research* 171 (1): 85–106.

Bo, Vincenzo, Marco Bortolini, Enrico Malaguti, Michele Monaci, Cristina Mora, and Paolo Paronuzzi. 2021. "Models and algorithms for integrated production and distribution problems." *Computers & Industrial Engineering* 154: 107003.

Campana, Giampaolo, Mattia Mele, and Barbara Cimatti. 2020. "Sustainable Optimisation of a Carousel for Foundry Processes." *Procedia Manufacturing* 43: 650–657.

Cleary, Paul, Joseph Ha, Vladimir Alguine, and Thang Nguyen. 2002. "Flow modelling in casting processes." *Applied Mathematical Modelling* 26 (2): 171–190.

Dolgui, Alexandre, Brigitte Finel, Nikolai N Guschinsky, Genrikh M Levin, and Francois B Vernadat. 2006. "MIP approach to balancing transfer lines with blocks of parallel operations." *IIE transactions* 38 (10): 869–882.

Dolgui, Alexandre, Nikolai Guschinsky, Genrikh Levin, and J-M Proth. 2008. "Optimisation of multi-position machines and transfer lines." *European Journal of Operational Research* 185 (3): 1375–1389.

Dolgui, Alexandre, NN Guschinsky, and GM Levin. 2009. "Graph approach for optimal design of transfer machine with rotary table." *International Journal of Production Research* 47 (2): 321–341.

Feo, Thomas A., and Mauricio G. C. Resende. 1995. "Greedy randomized adaptive search procedures." *Journal of global optimization* 6 (2): 109–133.

Foerster, Hildegard, and Gerhard Wascher. 2000. "Pattern reduction in one-dimensional cutting stock problems." *International journal of production research* 38 (7): 1657–1676.

Gilmore, Paul C., and Ralph E. Gomory. 1963. "A linear programming approach to the cutting stock problem—Part II." *Operations research* 11 (6): 863–888.

Ha, Joseph, Paul Cleary, Vladimir Alguine, and Thang Nguyen. 1999. "Simulation of Die Filling in Gravity Die Casting Using SPH and MAGMAsoft." *2nd International Conference on CFD in the Minerals and Process Industries* (December): 423–428.

Haessler, Robert W. 1975. "Controlling cutting pattern changes in one-dimensional trim problems." *Operations Research* 23 (3): 483–493.

Huang, Jing, Gürsel A. Süer, and Shravan B. R. Urs. 2012. "Genetic algorithm for rotary machine scheduling with dependent processing times." *Journal of Intelligent Manufacturing* 23 (5): 1931–1948.

Kamal, M.R.M., H.A.M. Tahir, M.S. Salleh, H.N.H. Bazri, M.K. Musa, and N.F. Bazilah. 2018. "Design and Thermal Analysis Simulation of Gravity Die Casting on Aluminium Alloy (ADC12)." *Journal of Advanced Manufacturing Technology (JAMT)* 12 (1 (2)): 271–288.

Malaguti, Enrico, Rosa Medina Durán, and Paolo Toth. 2014. "Approaches to real world two-dimensional cutting problems." *Omega* 47: 99–115.

Malaguti, Enrico, Rosa Medina Durán, and Paolo Toth. 2016. "A metaheuristic framework for Nonlinear Capacitated Covering Problems." *Optimization Letters* 10 (1): 169–180.

Martello, Silvano, and Paolo Toth. 1990. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc.

Richter, A., and J. Ondrášek. 2017. "Rotary Table Machine Input Parameters Optimization." In *Advances in Mechanism Design II*, edited by Jaroslav Beran, Martin Bílek, and Petr Žabka, Vol. 44, Cham, 399–405. Springer International Publishing.

Soylu, Banu, Ö. Kirca, and Meral Azizoğlu. 2007. "Flow shop-sequencing problem with synchronous transfers and makespan minimization." *International Journal of Production Research* 45 (15): 3311–3331.

Vanderbeck, François. 2000. "Exact algorithm for minimising the number of setups in the one-dimensional cutting stock problem." *Operations Research* 48 (6): 915–926.

Vijayaram, T.R., S. Sulaiman, A.M.S. Hamouda, and M.H.M. Ahmad. 2006. "Numerical simulation of casting solidification in permanent metallic molds." *Journal of Materials Processing Technology* 178 (1-3): 29–33.

Waldherr, Stefan, and Sigrid Knust. 2015. "Complexity results for flow shop problems with synchronous movement." *European Journal of Operational Research* 242 (1): 34–44.