Les Cahiers du GERAD ISSN: 0711–2440

The carousel scheduling problem

B.J. de Sousa Pessoa, D. Aloise,
L. dos Anjos Formiga Cabral

G–2017–37

May 2017

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

The carousel scheduling problem

Bruno Jefferson de Sousa Pessoa [a]
Daniel Aloise [b,c]
Lucídio dos Anjos Formiga Cabral [a]

[a] *Department of Scientific Computing, Federal University of Paraíba (João Pessoa), Brazil*

[b] *Département de Génie Informatique et Génie Logiciel, Polytechnique Montréal, Montréal (Québec) Canada*

[c] *Groupe d'études et de de recherche en analyse des décisions (GERAD), Montréal (Québec) Canada*

bruno@ci.ufpb.br
daniel.aloise@polymtl.ca
lucidio@ci.ufpb.br

May 2017

Les Cahiers du GERAD
G–2017–37

Abstract: Scheduling problems on which constraints are imposed with regard to the temporal distances between successive executions of the same task have numerous applications, ranging from task scheduling in real-time systems to automobile production on a mixed-model assembly line. This paper introduces a new NP-hard optimization problem belonging to this class of problems, namely the Carousel Scheduling Problem (CSP). We present a mathematical formulation for the CSP based on mixed-integer linear programming (MILP) as well as a series of cuts to improve its resolution via exact methods. Finally, we propose an iterative solution method which greatly reduces the number of variables in the CSP formulation. The reported computational experiments show that, for a given time horizon, the proposed iterative method increases the size of CSP instances solved up to optimality.

Keywords: scheduling, fair sequences, integer programming

Acknowledgments: Research of the second author was partially funded by CNPq-Brazil grant number 308887/2014-0

1 Introduction

Scheduling problems concern the allocation of tasks to limited resources over time [2]. Their importance has increased with industrial development and the advancement of manufacturing processes. Thus, they constitute one of the most important classes of problems addressed in Operations Research.

Among the various types of scheduling problems, some focus on the proportional sharing of resources over time. This type of scheduling problem brings a number of benefits, ranging from cost minimization in an assembly line to task scheduling in real-time systems. Toyota was one of the first companies to realize the advantages of producing different kinds of products, keeping the usage rate of all parts used by the assembly line as constant as possible. The company created the Just-in-Time production system, whose aim is to produce only the needy products in demanded quantities at the right time [8]. Toyota has used this system on its assembly lines, and as a result, storage costs and shortages have been reduced, and machine idleness has been minimized during the production process [18].

Scheduling problems where the temporal distances between successive executions of tasks or activities are not longer than a pre-specified distance are often referred to in the literature as distance-constrained scheduling problems [14] or, simply, fair sequences [18]. In this paper, we present a new optimization problem, namely the Carousel Scheduling Problem (CSP), which belongs to the class of problems that deals with fair sequences. More specifically, the CSP is a periodic scheduling problem with a finite time horizon, that given a set of tasks with different priorities and number of executions, aims to build a sequence of tasks such that the maximum temporal distances between successive executions of identical tasks are minimized. When taken together, these features provide the CSP with the necessary requirements to be regarded as a new scheduling problem.

Mathematically, the CSP can be defined as follows. Let $X = \{x_1, x_2, \ldots, x_n\}$ be a set of n symbols, each having unit length, and $c : X \to \mathbb{Z}^+$ a priority function. For a symbol x_i, we denote $c_i = c(x_i)$ its priority. Consider a sequence $S = s_1, s_2, \ldots, s_T$ of T symbols of X, with $T \leq TMAX$, where $TMAX$ is a parameter of the problem corresponding to the maximum length of the sequence. The frequency or the number of copies or occurrences of the symbols in S is a variable of the problem and is at least equal to one. The distance $d_{i,k,k+1}$ between the copies k and $k+1$ of symbol $x_i \in X$ is the number of symbols between them plus one. Assuming that l is the index of the last copy of symbol x_i in the sequence, $d_{i,l,1}$ is the distance between the last copy of x_i in a cycle and its first copy in the subsequent cycle. By denoting D_i the largest distance between all the consecutive copies of symbol x_i, including $d_{i,l,1}$, the goal is to find a sequence S that minimizes the largest product $D_i c_i$, for $i = 1, \ldots, n$.

Figure 1 illustrates a CSP solution for a set of symbols $X = \{A, B, C, D, E, F\}$ with priorities $c_A > c_B = c_C = c_D > c_E > c_F$. Two possible solution sequences for this instance are presented. They represent two feasible solutions with different distributions of symbols. In Solution 1, the copies of a given symbol are positioned one after the other, which results in a bad solution to the CSP. In the second solution, the copies are spaced as evenly as possible. Comparing these solutions on the basis of the distances between the copies of symbol A, the largest distance is observed to decrease from $D_A = 9$ in Solution 1 to $D_A = 3$ in Solution 2. Consequently, the product $D_A c_A$ also decreases. As the largest distances of all the symbols are minimized, Solution 2 is better than Solution 1.

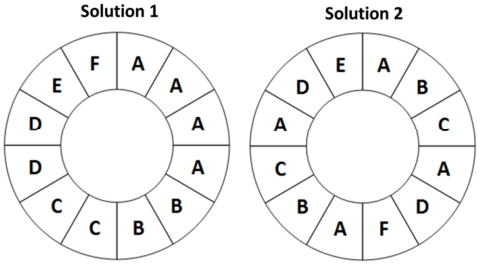

Figure 1: Uniform and non-uniform distribution of symbols.

It is important to note that in addition to bridging a gap in the literature, the CSP covers many specific applications that have been neglected . We discuss below some scenarios involving the CSP.

In a digital TV system, part of the bandwidth allocated to the TV stations is reserved for the airing of interactive applications, i.e., computer programs capable of providing interactivity between viewers and television content [20]. In addition to enhancing the TV watching experience, such programs facilitate the advertisement of products in a more dynamic manner. They are aired cyclically from servers to users on a one-way communication channel. To download them, the user terminals must listen to the communication channel until the arrival of the requested data. A cycle or sequence of transmissions consists of an ordered set of applications, which may be aired several times within a cycle in accordance with the application priority. At the end of the airing of the last application in the cycle, the first one is aired again. Considering that applications whose advertisers pay more for their placement are aired more frequently than others and that the optimal frequency of each application is also a variable of the problem, the objective is to minimize the maximum temporal distance between successive transmissions of the same application, and therefore the waiting time of the users. The length of the transmission sequence is variable, finite, and limited to a maximum value, which usually corresponds to the length of a TV program, a program block, or the time reserved for commercial advertisements. In general, the CSP occurs in most push system applications [4], environments involving data broadcast [16, 17], and it can also be applied to the traditional advertisement scheduling approach [10].

Periodic machine maintenance [1, 3] is another context in which the CSP can be applied. In such a scenario, a maintenance schedule should be designed to minimize the time between two successive maintenance services, given that the maintenance costs increase with the time elapsed since the last maintenance. As costs vary from machine to machine, the ideal frequency of maintenance services for each machine is a variable of the problem. Moreover, in most real-world situations, there is a limitation that the time horizon in which maintenance services are to be performed must be finite and in accordance with a schedule established a priori (e.g., one day, one week).

Real-time scheduling is also within the scope of the CSP [21]. Some real-time systems use a task scheduling mechanism based on the so-called proportional-share control, with the goal of minimizing the response time to user requests. For this purpose, the tasks should be executed at a uniform rate over time, through a prioritization strategy that aims to reflect the degree of importance of each task. As in the previous examples, the processes with higher priorities are executed at a higher frequency than others. To avoid starvation, i.e., when low-priority processes take too long to be executed, it is necessary to limit the maximum length of the sequences or cycles and establish the minimum frequency or number of executions for each of the tasks.

The remaining of this paper is organized as follows. Section 2 reviews the main studies related to the CSP and highlights its characteristics. Section 3 presents the mathematical formulation of the problem as a mixed-integer linear program. Section 4 proves that the problem is NP-hard. Section 5 presents some valid cuts to the CSP. In Section 6, our solution method based on exact algorithms is described in detail. Computational experiments are reported in Section 7. Finally, the conclusions are presented in Section 8.

2 Literature review

Fair sequences or constrained-distance scheduling problems have been mainly addressed in the fields of computer science and operations research. In the field of computer science, many studies have explored the development of theories and algorithms for scheduling real-time systems and problems arising from broadcast data transmission.

In 1989, Holte at al [15] formalized a scheduling problem of sporadic tasks, namely the *pinwheel problem*, inspired by satellite communication with a single ground station. Satellites send information periodically to the ground station, which has the ability to process data from a single satellite at a time. The transmission frequency is a satellite requirement and hence an input of the problem. Thus, the problem is to build a sequence of transmissions that meets the requirements of each satellite while simultaneously avoiding loss

of information, considering the processing restrictions of the ground station. A few years later, the *stride scheduling algorithm* was developed with the aim of implementing a proportional-share control over processor time [21]. To this end, the algorithm designs a schedule whose processes are executed evenly in time according to its priorities, and as a result, the response time of each process is reduced. Similarly, Baruah et al. [5] developed the notion of fairness in resource allocation, which they referred to as *P-fairness*. The underlying concept is to gradually schedule tasks according to weights calculated from their period and execution time. Accordingly, the tasks are scheduled with a certain uniformity over time. By the same time, Han et al. [14] introduced the *distance-constrained scheduling problem*, which involves scheduling a set of tasks to be executed periodically under the constraint that the temporal distance between successive executions should be less than a preset value. The pinwheel problem can be considered as its discrete version. Bar-Noy and Ladner [4] defined the *window scheduling problem*, which occurs in some applications involving broadcast data, such as push systems and on-demand media. Given a set of transmission channels, a set of pages to be transmitted in discrete time slots, and a time window associated with each page, a schedule that aims to solves the problem assigns pages to slots such that the temporal distance between any consecutive occurrences of the same page, considering all the transmission channels, is at most equal to its corresponding window. The *data broadcast problem* [16] can be regarded as the optimization version of the *windows scheduling problem*. It aims to find a sequence of message transmissions within an infinite time horizon, which minimizes the temporal distance between two consecutive transmissions of a message and thus the user waiting time. More recently, Garcia-Villoria and Salhi [10] introduced a new scheduling problem that occurs in the TV sector. It involves scheduling of commercial advertisements in order to satisfy advertisers and, therefore, broadcasters. The advertisers buy discrete time slots for advertisements and request that the successive transmissions of the same advertisement should be as evenly spaced in time as possible. The slots have airdates and are divided into three groups according to the audience rating (low, medium, and high). The number of times that the advertisements should be aired and the minimum number of slots to be allocated to the middle and high audience groups are inputs of the problem. The objective of the problem is to find a feasible schedule that minimizes the irregularity of the time intervals between two consecutive transmissions of the same advertisement.

In the field of operations research, the topic under discussion gained momentum with the introduction of the Just-in-Time production method, which was applied by Toyota to assembly lines in the 1980s. The Just-in-Time method produces different models at a uniform rate over time in order to reduce inventory costs [18]. By studying the Just-in-Time production system, Miltenberg [19] formulated it as a non-linear integer programming problem, aiming to minimize the total deviation between the obtained production rates and the desired production rates for each product. Anily et al. [1] studied an activity scheduling problem of various types within an infinite time horizon, which they referred to as *scheduling of maintenance services*. In their work, they considered a set of machines to be maintained with costs that increase in proportion to the elapsed time since the last maintenance. The larger the temporal distance, the higher is the cost involved. Hence, the goal of the problem is to find an optimal schedule that minimizes the average cost per unit time to maintain several different machines over time. Bar-Noy et al. [3] studied a generalized version of the *maintenance scheduling problem* by including the possibility of maintaining multiple machines in the same time slot. Corominas et al. [6] presented the *Response Time Variability Problem* (RTVP), which is a scheduling problem that arises whenever tasks (clients or products) need be sequenced in order to reduce the variability in the temporal distance between their executions [7, 11, 9]. In the RTVP, the task sequences are periodic and finite. Moreover, the demand or number of occurrences of each task is specified a priori; thus, it is an input to the problem. Garcia-Villoria and Pastor [12] introduced the minmax response time problem (mRTP), which is the minmax version of the RTVP.

Most of the problems cited above occur in a context where the time horizon considered is infinite. However, many practical situations require the sequence of activities to be performed in a range of finite and predetermined time, as discussed in the previous section. When considering a finite time horizon, it is necessary to treat the temporal distances between cycles, which have a direct impact on the definitions, formulations, and the solution methods developed to the problems. Among the problems mentioned, the only ones besides the CSP that treat periodic scheduling with a finite time horizon are the RTVP and mRTP. Although the

TV advertisement scheduling problem is also considered with a finite time horizon, it differs from the other problems in that it is not cyclical . Unlike the CSP, the lengths of the sequences in the RTVP and mRTP are established a priori based on the number of copies of each task, which is an input to the problem. Thus, a number of applications in which the frequency of each task cannot be determined in principle are not covered. In these practical situations, such frequencies are variables of the problem and not inputs. Thus, the CSP can be regarded as a generalization of the RTVP and mRTP.

3 Problem formulation

The terminology used in this section follows the formal definition of the problem presented in Section 1. The proposed formulation, based on MILP, is divided into two parts. The first part describes the problem data, its basic properties, and the strategy for calculating the distances between consecutive copies in the same cycle. The second part focuses on the description of variables and constraints involved in the calculation for copies belonging to consecutive different cycles.

Data	
n	Number of symbols.
X	Set of symbols $x_1, \ldots, x_n$.
M_i	Maximum number of copies of symbol $x_i \in X$.
K_i	Index set of the copies of $x_i \in X$, i.e., $K_i = \{1, \ldots, M_i\}$.
$TMAX$	Maximum length of the feasible sequences.
c_i	Priority of symbol $x_i \in X$.
H_{ik}	Set of positions that can be occupied by the k-th copy of symbol $x_i \in X$, i.e., $H_{ik} = \{k, \ldots, TMAX\}$.
Variables	
y_{ikh}	1 if the k-th copy of symbol $x_i \in X$ is in position $h \in H_{ik}$; 0 otherwise.
$d_{i,k,k+1}$	Distance between the k-th and $(k+1)$-th copies of symbol $x_i \in X$; $k \in \backslash\{M_i\}$.
D_i	Largest distance between two consecutive copies of symbol $x_i \in X$.
P	Largest product $D_i c_i$, $x_i \in X$.
p_{ik}	Position of the k-th copy of symbol $x_i \in X$.

Model

$$\text{Minimize} \quad P \tag{1}$$

$$\text{s.t.} \quad P \geq D_i c_i, \qquad \forall i = 1, \ldots, n, \tag{2}$$

$$D_i \geq d_{i,k,k+1}, \qquad \forall i = 1, \ldots, n, \forall k \in K_i \setminus \{M_i\}, \tag{3}$$

$$\sum_{i:x_i \in X} \sum_{\substack{k \in K_i \\ h \in H_{ik}}} y_{ikh} \leq 1, \qquad \forall h \in \{1, \ldots, TMAX\}, \tag{4}$$

$$\sum_{h \in H_{ik}} y_{ikh} \leq 1, \qquad \forall i = 1, \ldots, n, \forall k \in K_i, \tag{5}$$

$$\sum_{k \in K_i} \sum_{h \in H_{ik}} y_{ikh} \geq 1, \qquad \forall i = 1, \ldots, n, \tag{6}$$

$$p_{ik} = \sum_{h \in H_{ik}} y_{ikh} h, \qquad \forall i = 1, \ldots, n, \tag{7}$$

$$\sum_{h \in H_{ik}} y_{ikh} \geq \sum_{h \in H_{i,k+1}} y_{i,k+1,h}, \qquad \forall i = 1, \ldots, n, \forall k \in K_i \setminus \{M_i\}, \tag{8}$$

$$p_{i,k+1} \geq p_{ik} - \left(1 - \sum_{h \in H_{i,k+1}} y_{i,k+1,h}\right) TMAX, \qquad \forall i = 1, \ldots, n, \forall k \in K_i \setminus \{M_i\}, \tag{9}$$

$$d_{i,k,k+1} \geq p_{i,k+1} - p_{ik}, \qquad \forall i = 1, \ldots, n, \forall k \in K_i \setminus \{M_i\}, \tag{10}$$

$$\sum_{j:x_j \in X} \sum_{k \in K_j} \sum_{h \in H_{jk}} y_{jkh} \leq TMAX, \tag{11}$$

$$P, D_i, p_{ik} \geq 0, \qquad \forall i = 1, \ldots, n, \forall k \in K_i, \tag{12}$$

$$d_{i,k,k+1} \geq 0, \qquad \forall k \in K_i \setminus \{M_i\}, \tag{13}$$

$$y_{ikh} \in \{0, 1\}, \qquad \forall x_i \in X, \forall k \in K_i, \forall h \in H_{ik}. \tag{14}$$

The objective function (1) minimizes the maximum product between the largest distance of consecutive copies of symbol x_i and its priority c_i, $\forall x_i \in X$, defined by constraints (2). Constraints (3) establish, $\forall x_i \in X$, the largest distance between two consecutive copies of symbol x_i, considering only the distances belonging to a single cycle. Constraints (4), (5), and (6) guarantee, respectively, that at most one copy will be allocated at each position of the feasible sequences, each copy will be allocated no more than once, and each symbol will have at least one copy in the sequence. Constraints (7) calculate the positions of the copies of all the symbols. The unallocated copies are positioned out of the sequence, specifically at the zero position. Further, constraints (8) impose the condition that the $(k+1)$-th copy of symbol x_i cannot be allocated if the k-th copy is not allocated. Constraints (9) state that the positions of the allocated copies are defined in ascending order of k, i.e., $p_{i,k+1} \geq p_{ik}$. When the $(k+1)$-th copy is not allocated, the subtraction in the parentheses is equal to one, causing the right side of the inequality to be at most equal to zero, i.e., when $p_{ik} = TMAX$. Thus, the inequality is always valid when a given copy is not allocated. The calculation of the distances between consecutive copies of each symbol is described by constraints (10), while constraint (11) ensures that the maximum length of the feasible sequences is less than or equal to $TMAX$. Finally, constraints (12)–(14) are the domain constraints of the variables of the model.

According to the CSP definition, the number of copies of each symbol in the sequence is not known a priori, which makes the definition of the maximum number of allowed of copies of each symbol, i.e., M_i, for $i = 1, \ldots, n$ a step in the model decision process. It these values are overestimated, some copies are not used, i.e., they are allocated to the available positions.

To illustrate the relationship between the main components of the model, consider the set of symbols $X = \{1, 2, 3, 4, 5\}$, with priorities $c_1 = 10, c_2 = 9, c_3 = 8, c_4 = 7, c_5 = 6, M_i = 5, \forall x_i \in X$, and $TMAX = 15$. An optimal solution to this instance is shown in Figure 2, highlighting the distances between the copies of symbol 1.

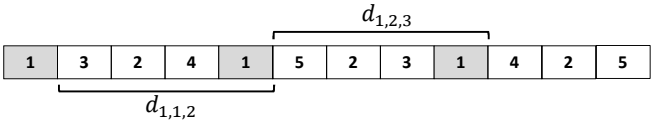

Figure 2: Optimal sequence for the set of symbols $X = \{1, 2, 3, 4, 5\}$, with priorities $c_1 = 10, c_2 = 9, c_3 = 8, c_4 = 7, c_5 = 6$, maximum number of copies $M_i = 5, \forall x_i \in X$, and $TMAX = 15$.

Since the number of allocated copies is less than the maximum for all the symbols, the position of some copies is zero. Hence, the copies of symbol 1, in ascending order, occupy the positions $1, 5, 9, 0, 0$, respectively. Note that the distances treated in the first part of the model are limited to a single cycle, and the distances between the last copy of each symbol of a cycle and the first of the next cycle are not considered. Such distances between cycles are part of the CSP definition because they are responsible for the cyclical nature of the problem. Figure 3 shows the optimal solution of Figure 2, emphasizing the notion of cycles and the distance between copies belonging to different consecutive cycles.

Variables and constraints involved in calculating the distances between copies from different cycles are described below.

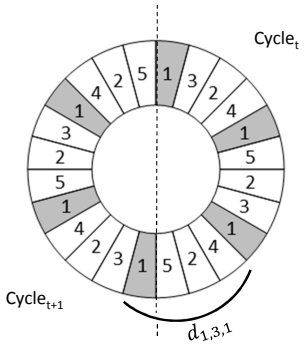

Figure 3: Notion of cycles and distance between the last copy of symbol 1 of cycle t and the first of cycle $t+1$.

W_i	Distance between the last allocated copy of a cycle and the first copy of the next cycle of symbol $x_i \in X$.
U_i	Sequence length plus the distance from the beginning of the sequence to the first copy of symbol $x_i \in X$.
R_i	Distance between the beginning of the sequence and the last allocated copy of symbol $x_i \in X$.
α_{ik}, w_{ik}	Auxiliary variables for finding the last unallocated copy of symbol $x_i \in X$.

$$D_i \geq W_i, \qquad \forall i = 1, \ldots, n, \tag{15}$$

$$W_i = U_i - R_i, \qquad \forall i = 1, \ldots, n, \tag{16}$$

$$U_i = \sum_{j:x_j \in X} \sum_{k \in K_j} \sum_{h \in H_{jk}} y_{jkh} + p_{i1}, \qquad \forall i = 1, \ldots, n, \tag{17}$$

$$R_i = \sum_{k \in K_i} \alpha_{ik}, \qquad \forall i = 1, \ldots, n, \tag{18}$$

$$\sum_{k \in K_i} w_{ik} = 1, \qquad \forall i = 1, \ldots, n, \tag{19}$$

$$\alpha_{ik} \leq 1 - w_{ik} + p_{ik}, \qquad \forall i = 1, \ldots, n, \forall k \in K_i, \tag{20}$$

$$\alpha_{ik} \leq w_{ik} TMAX, \qquad \forall i = 1, \ldots, n, \forall k \in K_i, \tag{21}$$

$$W_i, U_i, R_i, \alpha_{ik} \geq 0, \qquad \forall i = 1, \ldots, n, \forall k \in K_i, \tag{22}$$

$$w_{ik} \in \{0, 1\}, \qquad \forall i = 1, \ldots, n, \forall k \in K_i. \tag{23}$$

Constraints (15) include, in the set of distances between consecutive copies of symbols, the distances between the last allocated copy from one cycle and the first of the next cycle, which are calculated by means of constraints (16)–(21). The rationale behind it is presented in constraints (16), where U_i is the sum of the total length of the sequence to be constructed and the distance between the beginning of the sequence and the first copy of symbol x_i, and R_i is the distance between the beginning of the sequence and the last copy of x_i. Figure 4 shows the representation of the sequence 2-1-5-3-1-5, emphasizing the subsequences corresponding to variables $U_1, R_1,$ and W_1 in parts (a), (b), and (c), respectively. Variable U_i can be easily calculated by means of constraints (17), whereas R_i is the major obstacle to finding W_i, because the position of the last allocated copy of x_i is not known initially. Constraints (18)–(21) calculate the distance between the beginning of the sequence and the last allocated copy of symbol $x_i, \forall x_i \in X$. They ensure that $R_i = \sum_{k \in K_i} \alpha_{ik}$ is less than or equal to the positions of the copies of x_i. Since the model minimizes $W_i, \forall x_i \in X$, R_i is maximized, i.e., it is raised to the highest position of a copy of x_i (recall that unallocated copies have positions set to zero). Non-negativity and integrality constraints on the problem variables, i.e., (22) and (23), respectively, complete the model.

The large number of variables of the model is mainly due to the maximum number of copies of each symbol, which constitutes the index set of several variables. The variables $y_{ikh}, p_{ik}, d_{i,k,k+1}, \alpha_{ik}, w_{ik}$, for $i = 1, \ldots, n, k \in K_i,$ and $h \in H_{ik}$, depend on the maximum number of copies M_i, as the index k is directly linked to it. In Section 5 we present a mathematical programming model that yields good upper bounds for

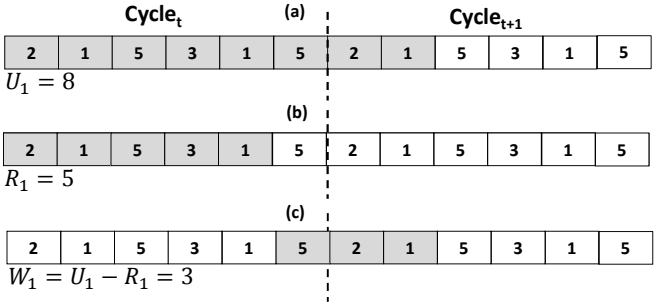

Figure 4: Representation of the sequence 2-1-5-3-1-5 with emphasis on the subsequences corresponding to variables U_1 (a), R_1 (b), and W_1 (c).

the maximum number of used copies of each symbol, enabling that optimal solutions are obtained via exact methods in much smaller computing times.

4 NP-hardness of the CSP

In this section, we show that the decision version of the CSP is NP-complete. The reduction is obtained from the periodic maintenance scheduling problem [3], which can be defined as follows. Given n machines and service intervals $l_1, l_2, \ldots, l_n$ such that $\sum_{i=1}^{n} \frac{1}{l_i} \leq 1$, does there exist an infinite maintenance service schedule, in which consecutive services of machine i are spaced by exactly l_i time slots and no more than one machine is serviced in a single time slot?

To align the two problems more closely, consider the decision version of the CSP. I.e., Given the set of symbols $X = \{x_1, \ldots, x_n\}$ each having unit length, and priorities $c_1, c_2, \ldots, c_n \in \mathbb{Z}^+$, does there exist a finite sequence of symbols whose largest product $P_i = c_i D_i$, associated to each $x_i \in X$, is smaller or equal to B? We show that a positive or negative answer to one of the questions above necessarily implies the same answer to the other for a particular value of B. The following propositions form the foundation of the main result of this section.

Proposition 1 *Let S^* be a CSP solution sequence of length T, and m_i the number of used copies of symbol $x_i \in X$ in S^*. The inequalities $T \leq D_i m_i$ always hold, for $i = 1, \ldots, n$.*

Proof. Let $d_{i,m_i,1}$ be the distance between the last allocated copy of symbol x_i of a cycle and the first copy of x_i in the next cycle. Assume that the distance between consecutive copies of x_i is $D_i = \max\{\max_{k=1,\ldots,m_i-1}\{d_{i,k,k+1}\}, d_{i,m_i,1}\}$. Then,

$$T = \sum_{k=1}^{m_i-1} d_{i,k,k+1} + d_{i,m_i,1} \leq \sum_{k=1}^{m_i} D_i = D_i m_i.$$

□

Corollary 1 *If $T \leq D_i m_i$, for $i = 1, \ldots, n$, then $\sum_{i=1}^{n} \frac{1}{D_i} \leq 1$.*

Proof. The inequalities to be analyzed are listed below:

$$\begin{aligned} T &\leq D_1 m_1, \\ T &\leq D_2 m_2, \\ &\vdots \\ T &\leq D_n m_n. \end{aligned}$$

Dividing both sides by their respective D_i, the inequalities become

$$\begin{aligned}\frac{T}{D_1} &\leq m_1,\\ \frac{T}{D_2} &\leq m_2,\\ &\vdots\\ \frac{T}{D_n} &\leq m_n.\end{aligned}$$

Therefore,

$$\begin{aligned}T\left(\sum_{i=1}^{n}\frac{1}{D_i}\right) &\leq T,\\ \sum_{i=1}^{n}\frac{1}{D_i} &\leq 1\end{aligned}$$

□

Proposition 2 *Let $L = \{l_1, l_2, \ldots, l_n\}$ be a set of distances, whose $lcm(l_1, \ldots, l_n)$ is equal to B and $\sum_{i=1}^{n}\frac{1}{l_i} = 1$. The optimal solution f of a CSP instance with symbols $X = \{x_1, \ldots, x_n\}$ and priorities $c_i = \frac{B}{l_i}$ is then greater than or equal to B.*

Proof. Let $D = \{D_1, \ldots, D_n\}$ be the set of the largest distances of a feasible CSP solution. If $f < B$, then $P_i = c_i D_i < B$, which implies $D_i < l_i$, for $i = 1, \ldots, n$. According to Corollary 1, $\sum_{i=1}^{n}\frac{1}{D_i} \leq 1$ is a necessary condition for the set D to be schedulable. Since $\sum_{i=1}^{n}\frac{1}{l_i} = 1$, if $D_i < l_i$, for any symbol $x_i \in X$, $\sum_{i=1}^{n}\frac{1}{D_i} > 1$, resulting in an unschedulable set of distances. Consequently, $P_i = c_i D_i \geq B$, for $i = 1, \ldots, n$. □

Proposition 3 *Consider the CSP instance of Proposition 2. If $f = B$, all the distances between consecutive copies of symbol x_i are exactly $D_i, \forall x_i \in X$.*

Proof. Based on Proposition 2, if $f = B$, then $P_i = B$, for $i = 1, \ldots, n$. Furthermore, $D_i = l_i$ and $\sum_{i=1}^{n}\frac{1}{D_i} = 1$. Let S be a the considered solution sequence to the given instance. Supposing that its length is T, a symbol $x_i \in X$ has at least $m_i = \left\lceil \frac{T}{D_i} \right\rceil$ copies in S; otherwise, its largest distance would be greater than D_i. Assume that the number of copies is given by $m_i = \frac{T}{D_i} + \lambda_i$, where λ_i, for $i = 1, \ldots, n$, is a non-negative real number, and $\sum_{i=1}^{n}(\frac{T}{D_i} + \lambda_i) = T$. Developing this equality we obtain

$$\left(\frac{T}{D_1} + \lambda_1\right) + \left(\frac{T}{D_2} + \lambda_2\right) + \cdots + \left(\frac{T}{D_n} + \lambda_n\right) = T.$$

Dividing both sides by T, we have

$$\begin{aligned}\left(\frac{1}{D_1} + \frac{\lambda_1}{T}\right) + \left(\frac{1}{D_2} + \frac{\lambda_2}{T}\right) + \cdots + \left(\frac{1}{D_n} + \frac{\lambda_n}{T}\right) &= 1,\\ \sum_{i=1}^{n}\frac{1}{D_i} + \sum_{i=1}^{n}\frac{\lambda_i}{T} &= 1.\end{aligned}$$

As $\sum_{i=1}^{n}\frac{1}{l_i} = 1$ and $D_i = l_i$, $\sum_{i=1}^{n}\frac{\lambda_i}{T} = 0$. Knowing that λ_i is a non-negative real number, the last equality holds if and only if $\lambda_i = 0$, FOR $i = 1, \ldots, n$. Consequently,

$$T = D_i m_i, \text{for } i = 1, \ldots, n. \tag{24}$$

Now, consider a symbol x_i in S. Since x_i has m_i copies, the set of distances between its consecutive copies has m_i elements: $d_{i,1,2}, \ldots, d_{i,m_{i-1},m_i}, d_{i,m_i,1}$. Because the sum of these distances is equal to T, if one of them is less than D_i, then $T < D_i m_i$, which contradicts (24). Thus, all the distances between consecutive copies of symbol x_i in $S, \forall x_i \in X$, must be exactly equal to D_i. □

Theorem 1 *The Carousel Scheduling Problem is NP-hard.*

Proof. Consider the following reduction from the periodic maintenance scheduling problem (PMSP). Given an instance of the PMSP with n machines and service intervals $l_1, \ldots, l_n$, such that $\sum_{i=1}^{n} \frac{1}{l_i} \leq 1$, the corresponding instance of the CSP has $(n+t)$ symbols, where $t\frac{1}{B} = 1 - \sum_{i=1}^{n} \frac{1}{l_i}$ and $B = lcm(l_1, \ldots, l_n)$, such that $\sum_{i=1}^{n} \frac{1}{l_i} + \sum_{i=n+1}^{n+t} \frac{1}{B} = 1$.The priorities of the symbols are given by $c_i = \frac{B}{l_i}, i = 1, \ldots, n$, and $c_j = 1$, for $j = n+1, \ldots, n+t$, and the maximum length of the feasible sequences, $TMAX$, is equal to B. We show that the PMSP has a solution if, and only if, the corresponding instance of the CSP has an objective value smaller or equal to B.

(If) If the PMSP has a solution to the above instance, the distance between consecutive maintenance services is exactly l_i, for $i = 1, \ldots, n$. If $\sum_{i=1}^{n} \frac{1}{l_i} = 1$, then $t = 0$, and any sequence of B symbols belonging to the infinite solution sequence is also a solution to the CSP, whose distances between consecutive copies are all equal to $l_i = \frac{B}{c_i}$ and the objective value is $P = c_i l_i = B$, for $i = 1, \ldots, n$. If $\sum_{i=1}^{n} \frac{1}{l_i} < 1$, there are t time slots not occupied by symbols in any interval of B slots. In this case, it suffices to place t distinct symbols in such slots to reach a solution sequence for the CSP, whose objective value is exactly B.

(Only If) First, Proposition 2 shows that the solution value of the given CSP instance cannot be smaller than B. Then, from Proposition 3, we can conclude that, if a solution to the considered CSP instance has an objective value equal to B, all the distances between the consecutive copies of a symbol x_i, for $i = 1, \ldots, n$, are equal to l_i. Further, if $t \geq 1$, the only distance between consecutive copies of a symbol $x_j, j = n+1, \ldots, n+t$, is equal to B, since there is only one copy of this symbol in the sequence. Accordingly, disregarding the symbols x_j, for $j = n+1, \ldots, t$, the sequence is a solution for the PMSP whose period is B. □

5 Cuts

This section presents valid cuts for the CSP that aim to reduce the search space of the problem and hence improve the performance of exact methods in searching for optimal solutions. The terminology used below is based on the mathematical formulation described in Section 3.

Proposition 4 *Let x_r and x_s be symbols of a CSP instance. There exists an optimal solution, where symbol x_s is placed in the first position of the solution sequence, i.e., $y_{s11} = 1$.*

Proof. Assume that the symbol placed in the first position of an optimal solution S^* is x_r, i.e., $y_{r11} = 1$, and the first copy of symbol x_s is in position $p_{s1} > 1$. Let us consider a sequence S', shifted to the right in relation to S^*, such that the k-th copy of symbol x_i, belonging to S^*, is placed in the position $p'_{ik} = (p_{ik} + \beta)\ mod\ T, \forall x_i \in X, \forall k \in K_i$, where $\beta = T - p_{s1} + 1$ and T is the length of S^*. That solution is also optimal to the considered instance with $p_{s1} = 1$. □

Proposition 5 *Consider a CSP instance with n symbols and such that $c_i \geq c_{i+1}$, for $i = 1, \ldots, n-1$. There exists an optimal solution to this instance in which $D_i \leq D_{i+1}$*

Proof. Let S^* be an optimal sequence to the given instance. Suppose, without loss of generality, that the objective value of S^* is $P^* = c_i D_i$, and there exists a symbol x_j in S^* such that $c_i > c_j$ and $D_i > D_j$. Exchanging the positions of symbols x_i and x_j in S^*, we have a sequence S' with objective value $P = \max\{c_i D_j, c_j D_i\} < c_i D_i = P^*$. Extending this result to the entire set of symbols, we obtain an optimal sequence where $D_i \leq D_{i+1}$, for $i = 1, \ldots, n-1$. □

Proposition 6 *Consider a CSP instance with $n \geq 2$ symbols. There exists an optimal sequence in which there are no copies of the same symbol placed in consecutive positions. Therefore, the constraints*

$$y_{ikh} + y_{i,k+1,h+1} \leq 1, \qquad \forall i = 1, \ldots, n, \forall k \in K \setminus \{M_i\}, \forall h \in (H_{ik} \cap H_{i,k+1}),$$

do not fully eliminate the set of optimal solutions of the CSP.

Proof. As $n \geq 2, D_i \geq 2$, for $i = 1, \ldots, n$, in any feasible solution. Let S^* be an optimal solution whose objective value is f^*. If there exist two copies of a symbol x_i placed in consecutive positions, where $D_i^* \geq 2$, one of them can be eliminated, generating a new sequence whose objective value is less than or equal to f^*. Proceeding in this way, we can also generate an optimal sequence, possibly smaller, where there are no consecutive copies of the same symbol placed in consecutive positions. □

The set of cuts presented in the sequel were observed to be valid in all our computational results presented in Section 7. Although intuitive, we were not able to provide a formal proof for it. For that reason, we present it as a conjecture for future research and reference.

Conjecture 1 *Consider a CSP instance with n symbols and such that $c_i \geq c_{i+1}$, for $i = 1, \ldots, n-1$. There exists at least one optimal solution to this instance such that $m_i \geq m_{i+1}$, where m_i is the number of used copies of symbol x_i, and such that*

$$\sum_{h \in H_{i+1,k}} y_{i+1,k,h} \leq \sum_{h \in H_{ik}} y_{ikh}, \qquad \forall x_i \in X \setminus \{x_n\}, \forall k \in K_{i+1}.$$

Indeed, one of the main difficulties in solving the CSP is to establish valid values for the maximum number of copies M_i of each symbol $x_i \in X$ so as to reduce the number of variables of the problem indexed by K_i. The maximum length of the feasible sequences is a trivial upper bound to those values. However, since such a value is used as an index of several variables, the total number of variables would be extremely large. Based on Proposition 6 and the imposition that each symbol must have at least one copy in the solution sequence, a slightly better value is $M_i = \min\{\frac{TMAX}{2}, TMAX - (n-1)\}$, for $i = 1, \ldots, n$. Nevertheless, our experiments showed that this is not enough to allow CSP resolution via exact methods. The result below concerns obtaining more accurate values for the maximum number of copies M_i of each symbol $x_i \in X$.

Proposition 7 *Let $X = \{x_1, \ldots, x_n\}$ be a set of symbols such that $c_1 \geq \cdots \geq c_n$. The maximum number of copies M_{i^*}, for some $x_i^* \in X$, can be made equal to ρ_{i^*}, where*

$$\rho_{i^*} = \textit{Maximize} \quad m_{i^*} \tag{25}$$

$$\begin{aligned}
s.t. \quad P &\geq c_i D_i, && \forall i = 1, \ldots, n, && (26)\\
D_i &\leq D_{i+1}, && \forall i = 1, \ldots, n-1, && (27)\\
T &\leq m_i D_i, && \forall i = 1, \ldots, n, && (28)\\
T &= \sum_{i \in X} m_i, && \forall i = 1, \ldots, n, && (29)\\
m_i &\geq 1, && \forall i = 1, \ldots, n, && (30)\\
T &\leq TMAX, && && (31)\\
P, D_i, T, m_i &\geq 0, && \forall i = 1, \ldots, n. && (32)
\end{aligned}$$

Proof. The objective function (25) aims to maximize the number of copies of symbol $x_i^* \in X$. Constraints (26) make P larger than all the products between a symbol priority and the largest distance between two of its consecutive copies. Constraints (27) and (28) consist of the results presented in Propositions 5 and 1, respectively. Constraints (29) state that T is equal to the total number of copies used by the whole set of

symbols, while constraints (30) impose that at least one copy of each symbol be used. Let Q be the polyhedron related to constraints (26)–(32). Since constraints (26)–(32) are all valid for the CSP, every solution for the CSP is feasible in Q. Consequently, a projection of Q onto the subspace defined by the variables of the CSP is a relaxation for it. □

The proof above also allows to conclude that if (25) is replaced by (1), a lower bound for the CSP is obtained. Although model (25)–(32) is able to provide valid values for the maximum number of copies of each symbol $x_i \in X$, there is no guarantee of their quality. Indeed, the use of an upper bound solution value $\overline{f}$ for the CSP problem can improve the quality of the values obtained by the model. For that purpose, it suffices to add the inequality

$$P \leq \overline{f}. \tag{33}$$

The closer $\overline{f}$ is with respect to the CSP optimal solution, the better the accuracy achieved for the maximum number of copies M_i of each symbol $x_i \in X$. Model (25)–(33) is called MNC_{i^*} (maximum number of copies) thereafter in the text, where i^* stands for the index of the target symbol.

6 Solution method

Algorithm 1 describes our iterative exact method to solve the CSP. The input parameters for the algorithm are: X, the set of symbols, $c : X \leftarrow \mathbb{Z}^+$, the priority function, $TMAX$, the maximum length of the sequence, a real value Δ, $0 < \Delta < 1$, and it_{max} that represents the maximum number of allowed iterations of the algorithm. The idea behind the algorithm is to solve the CSP model, described in Section 3, with much less variables indexed by M_i, for $i = 1, \ldots, n$.

Algorithm 1 Exact method for the CSP problem

```
1: procedure CSPSolver(X, c, TMAX, Δ, it_max)
2:     LB ← value of (1) subject to (26)–(32)
3:     f̄ ← LB + LB × Δ
4:     for it = 1 to it_max do
5:         for i = 1 to n do
6:             M_i ← result of MNC_i
7:         end for
8:         solve CSP with cut P ≤ f̄
9:         if (the optimal solution was found) exit
10:        f̄ ← f̄ + LB × Δ
11:        it ← it + 1
12:    end for
13: end procedure
```

Initially, a valid lower bound LB is obtained in line 2 and an initial estimate for the CSP solution $\overline{f}$ is calculated in line 3. Then, the loop of lines 4-12 is repeated for at most it_{max} iterations. In the inner loop of lines 5-7, the maximum number of copies M_i is calculated for each symbol $x_i \in X$, with $\overline{f}$ as the current value for inequality (33). Next, at line 8, the CSP model is solved using M_i as the maximum value in set K_i, for $i = 1, \ldots, n$, and cut $P \leq \overline{f}$ which can make the problem possibly infeasible. If that does not occur, the optimal CSP solution was found and the algorithm terminates. Otherwise, $\overline{f}$ is increased in line 10 and the iteration is incremented in line 11. The algorithm is guaranteed to find an optimal solution whenever it_{max} is set to $\left\lceil \frac{c_{max} \times n - LB}{LB \times \Delta} \right\rceil$, where c_{max} is the largest priority value among all symbols.

Indeed, iterations are each time longer to execute. This is due to the fact that as $\overline{f}$ increases, the values M_i likely increase too, which increases, in turn, the number of variables for the CSP model. Consequently, the value of Δ plays an important role in the algorithm. On the one hand, small Δ values may result in many iterations without changing considerably the M_i values to turn feasible the CSP instance. On the other hand, large Δ values may increase too much the M_i values making the CSP problem very expensive to compute in each iteration, and thus, directly affecting the performance of our algorithm.

7 Computational experiments

Our experiments are designed to assess the performance of Algorithm 1 on different instances. The MILP model was solved with CPLEX 12.6 as well as the NLP models solved in lines 2 and 6 of Algorithm 1 which were linearized via binary expansion of variables (see e.g.[13]). The experiments were carried out on an Intel Core I7 with 1.9 GHz and 6 GB of RAM, running Ubuntu 14.04 LTS operation system. The time limit for the execution of each instance was set to one hour of CPU time. All the reported computational times in the following tables are given in seconds.

First, we describe the instances used in our experiments. Then, we analyze the impact of the cuts presented in Section 5. Next, we discuss the results obtained from the execution of the iterative solution method of Section 6. Finally, the last set of experiments aims to compare the performance of the different exact approaches for the CSP solution.

7.1 Instances

We generated 90 instances grouped into 18 classes according to the number of symbols (n) and the maximum length of the sequences ($TMAX$). We considered instances with 5, 7, 9, 11, 13, and 15 symbols, and $TMAX$ equal to $2n, 3n,$ and $4n$. For each pair $(n, TMAX)$, five distinct instances were generated by assigning random priorities to each symbol in the interval $[1, 2n]$. The name of the classes follows the following format: $class\_n\_TMAX$. Thus, $class\_5\_15$ denotes the class of instances with five symbols and $TMAX = 15$. All the instances are available online at https://sites.google.com/site/carouselschedulingproblem/instances.

7.2 Impact of the cuts

In this section, we analyze the impact of the cuts presented in Section 5 on solving the CSP model of Section 3 directly via a MILP solver. These cuts are derived from Propositions 3, 4, 5, and Conjecture 1. They are named $c1$, $c2$, $c3$, and $c4$, respectively. The reported experiments evaluate the addition of one set of cuts at a time to the CSP model. Thus, each instance is solved four times, one for each set of cuts. We report results on classes of instances in which at least one of its instances is solved to optimality by the use of a set of cuts alone. Further, we do not present results for class ($class\_5\_10$) as its instances are solved too quickly.

In Table 1, **NC** is the sum of the maximum numbers of copies of each symbol, i.e., the trivial value $\lceil \frac{TMAX}{2} \rceil$ as calculated from Proposition 6. **CSP** refers to the CSP model presented in Section 3 without none of the proposed cuts, **CSP_c1** to **CSP_c4** concern the CSP model with the addition of cuts c1 to c4, individually, and **CSP_C** corresponds to the CSP model with the addition of all cuts. Column **Time** contains the average time to solve the instances of the classes considered. The label TL (Time Limit) is used whenever at least one of the instances of the class is not solved in one hour of allowed CPU time. Finally, the **Gap** column presents the average relative difference (in %) between the best avegare solution values and lower bounds reported at the end of CPLEX executions.

Table 1: Comparison of CSP models enhanced by the proposed cuts.

		CSP		CSP_c1		CSP_c2		CSP_c3		CSP_c4		CSP_C	
Class	**NC**	**Time**	**Gap(%)**	**Time**	**Gap(%)**	**Time**	**Gap(%)**	**Time**	**Gap(%)**	**Time**	**Gap(%)**	**Time**	**Gap(%)**
class_5_15	40	73.93	0.00	22.01	0.00	104.42	0.00	50.29	0.00	19.09	0.00	6.26	0.00
class_5_20	50	1161.71	0.00	330.73	0.00	1268.57	0.00	154.86	0.00	326.95	0.00	92.59	0.00
class_7_14	49	23.57	0.00	9.39	0.00	20.54	0.00	20.39	0.00	10.63	0.00	2.68	0.00
class_7_21	77	TL	10.63	TL	9.24	TL	11.44	TL	2.69	668.48	0.00	148.51	0.00
class_9_18	81	TL	4.63	TL	4.00	TL	2.87	TL	5.55	329.03	0.00	38.59	0.00
class_11_22	121	TL	11.87	TL	5.67	TL	10.74	TL	11.06	TL	2.11	148.32	0.00

The results show a significant improvement of model CSP_C over model CSP. The total computational time spent was considerably cut. If we consider only the classes for which all instances were solved to

optimality using both models, the average computing times were reduced by a factor of 11. Moreover, the joint use of the cuts in model CSP_C allowed to solve all the tested instances of the referred classes to optimality.

7.3 Solution method results

Algorithm 1 was tested on all instances with $\Delta = 0.05$ after limited computational experiments. Since the cuts discussed in the previous section improved the CSP performance, we included them into the CSP formulation solved within our solution method. All results presented in the current section correspond to average solution values obtained for the instances of each class.

In Table 2, **NC** denotes the sum of the maximum numbers of copies of each symbol obtained from model MNC, **LB** denotes the lower bound value LB obtained in line 2 of Algorithm 1, **BS** is the best solution found for the CSP by our iterative solution method, and **Gap** is the gap (in percentage) between the best solution and the lower bound found by CPLEX in the last iteration of the algorithm. This value is actually zero whenever the instance is solved to optimality. Column **LB Time** is the time spent on computing LB, **MNC Time** is the time spent on solving MNC, and **Total Time** is the total CPU time of Algorithm 1. The label TL is used whenever at least one instance of a class is not solved to optimality. Column **NE** corresponds to the number of times that the CSP formulation was solved in line 8 of Algorithm 1. Finally, **NS** indicates the number of instances not solved to optimality within the time limit of 1 hour, except for instances class_13_52 and class_15_60 for which the algorithm was allowed to run for five hours in order to obtain at least one CSP feasible solution in all instances of the class.

Table 2: Solution method results.

IC	NC	LB	BS	Gap %	LB Time	MNC Time	Total Time	NE	NS
class_5_10	10.20	39.60	39.60	0.00	0.22	0.28	0.65	1.00	0
class_5_15	16.20	32.40	34.40	0.00	0.32	0.32	2.00	1.60	0
class_5_20	24.00	29.00	32.20	0.00	0.14	0.54	30.78	2.20	0
class_7_14	14.00	55.40	55.40	0.00	0.28	0.47	0.84	1.00	0
class_7_21	22.20	61.20	62.40	0.00	0.42	0.99	7.04	1.00	0
class_7_28	33.40	63.20	64.80	0.00	0.40	2.05	150.24	1.00	0
class_9_18	19.80	115.60	115.60	0.00	0.33	1.01	1.64	1.00	0
class_9_27	30.40	101.40	101.40	0.00	0.37	3.36	35.33	1.00	0
class_9_36	44.60	87.00	88.20	0.00	0.35	4.26	448.68	1.00	0
class_11_22	24.40	161.40	161.40	0.00	0.25	1.74	3.43	1.00	0
class_11_33	38.20	139.20	139.20	0.00	0.36	5.91	49.00	1.00	0
class_11_44	52.60	149.00	150.40	0.87	0.55	8.74	TL	1.00	1
class_13_26	28.60	215.40	215.40	0.00	0.24	2.49	4.77	1.00	0
class_13_39	46.40	218.40	218.40	0.00	0.40	7.83	404.49	1.00	0
class_13_52	67.60	187.40	191.80	2.51	0.74	13.18	TL	1.20	2
class_15_30	33.60	336.20	336.20	0.00	0.20	4.66	8.58	1.00	0
class_15_45	56.40	301.20	302.20	0.31	0.64	10.07	TL	1.00	1
class_15_60	81.40	286.60	303.00	5.31	0.85	17.38	TL	1.00	5

From a total of 90 instances, 81 were solved to optimality within the allowed limit. Instances with up to 9 symbols were all optimally solved regardless of the $TMAX$ value. Further, our solution method solved to optimality 93.33%, 86.67%, and 60% of the instances with 11, 13, and 15 symbols, respectively. For $TMAX = 2n, 3n, 4n$, the algorithm was able to obtain optimal solutions in 100%, 96.67%, and 73.33% of the instances, respectively.

As expected, the performance of the solution method decreases rapidly with the increase of **NC** due to the exponential character of the branch-and-bound algorithm. Up to 50 copies, the solution method solved all instances. Above this, the solution method had difficulties to found optimal solutions. The maximum number of copies of an instance solved to optimality was 60.

The results showed that LB values are not far from the best solution values found, max. $\approx 11\%$ and on avegare 1.8%, which demonstrates the quality of the bounds obtained in line 2 of Algorithm 1. Consequently,

the big majority of the instances was solved with only one iteration of our solution method. More details about our experiments can be found at https://sites.google.com/site/carouselschedulingproblem/computational-experiments.

7.4 Comparison between the proposed methods

Our last set of experiments compares our solution method with the branch-and-cut algorithm used by CPLEX for solving MILP problems such as the CSP. Table 3 and 4 present results on the basis of three performance metrics: CPU times (**Time**), best CSP solution values (**BS**), and number of instances not solved to optimality in each class (**NS**). The **CSP_C** column refers to the solution via CPLEX of the CSP model enhanced by the cuts presented in Section 5, and **SM** refers to the CSP solution via our iterative solution method. Table 3 is restricted to the classes of instances for which all of its instances were solved to optimality by both methods in one hour.

Table 3: Average CPU times obtained by CPLEX on solving model CSP_C and those obtained by SM.

	Time	
IC	**CSP_C**	**SM**
class_5_10	**0.24**	0.65
class_5_15	6.26	**2.00**
class_5_20	92.59	**30.78**
class_7_14	2.68	**0.84**
class_7_21	148.51	**7.04**
class_9_18	38.59	**1.64**
class_9_27	802.90	**35.33**
class_11_22	148.32	**3.43**
class_13_26	1165.41	**4.77**
class_15_30	1242.89	**8.58**

We clearly notice in Table 3 that our iterative solution method **SM** outperforms CPLEX in general, except for the smallest used instances in *class*_5_10. For these instances, the number of variables in the CSP formulation is not large enough to justify the use of our iterative method. In contrast, the cutoff attains a factor of approximately 245 for class_13_26.

Table 4: BS and NS results obtained by CPLEX on solving CSP_C and SM.

	CSP_C		SM	
IC	**BS**	**NS**	**BS**	**NS**
class_7_28	65.20	3	64.80	0
class_9_36	98.00	5	88.20	0
class_11_33	142.00	4	139.20	0
class_11_44	179.00	5	150.40	1
class_13_39	246.80	5	218.40	0
class_13_52	223.20	5	191.80	2
class_15_45	379.00	5	302.20	1
class_15_60	388.80	5	303.00	5

Table 4 focus on the remaining class of instances. CPLEX and our iterative algorithm was allowed to run for five hours for the instances of the classes class_13_52 and class_15_60 so that it could obtain at least one feasible CSP solution. Regarding CSP solution values, those obtained by **SM** are on average approximately 12% better than those obtained via CPLEX, reaching a peak of 22% of improvement for the class class_15_60. In general, considering all classes of instances, **SM** allowed to solve more instances to optimality given the established time limit of one hour: 81 against 53 by CPLEX while directly solving model **CSP_C**.

8 Conclusions

In this paper, we introduced the Carousel Scheduling Problem (CSP), a new optimization problem that has a wide spectrum of applications, ranging from task scheduling in real-time systems to automobile production on a mixed-model assembly line. We showed that the CSP problem is NP-hard and presented its formulation based on mixed-integer linear programming. Since the number of variables of the problem increases rapidly with the input size, we developed cuts that reduce the search space of the problem by eliminating symmetries and feasible regions, but preserving at least one optimal solution. As a result, we significantly improved the efficiency of the formulation. In addition, we developed a mathematical model that provides good lower bounds for the problem. All these improvements were put together and resulted in an efficient iterative exact method to solve the CSP which outperforms its solution via CPLEX.

To carry out the experiments we created a set of benchmark instances that will be used as a reference in the future. Some instances could not be solved optimally within the time limit, which establishes a frontier to future works. In this way, future research may focus on designing heuristic and metaheuristic procedures as well as hybrid methods mixing exact and heuristic approaches. Moreover, exact methods can still be proposed over ours with the development of new branch-and-bound algorithms exploring the polytope of the CSP problem.

References

[1] Shoshana Anily, Celia A. Glass, and Refael Hassin. The scheduling of maintenance service. Discrete Applied Mathematics, 82(1):27–42, 1998.

[2] K.R. Baker. Introduction of Sequencing and Scheduling. Wiley, 1974.

[3] Amotz Bar-Noy, Randeep Bhatia, Joseph (Seffi) Naor, and Baruch Schieber. Minimizing service and operation costs of periodic scheduling. Math. Oper. Res., 27(3):518–544, August 2002.

[4] Amotz Bar-Noy and Richard E. Ladner. Windows scheduling problems for broadcast systems. SIAM J. Comput., 32(4):1091–1113, 2003.

[5] S. K. Baruah, N. K. Cohen, and D. A. Plaxton, C. G.and Varvel. Proportionate progress: A notion of fairness in resource allocation. Algorithmica, 15(6):600–625, 1996.

[6] Albert Corominas, Wieslaw Kubiak, and Natalia Moreno Palli. Response time variability. Journal of Scheduling, 10(2):97–110, 2007.

[7] Albert Corominas, Wieslaw Kubiak, and Rafael Pastor. Mathematical programming modeling of the response time variability problem. European Journal of Operational Research, 200(2):347–357, 2010.

[8] Tanka Nath Dhamala and Wieslaw Kubiak. A brief survey of just-in-time sequencing for mixed-model systems. International Journal of Operations Research, 2(2):38–47, 2005.

[9] Alberto García-Villoria, Albert Corominas, Xavier Delorme, Alexandre Dolgui, Wieslaw Kubiak, and Rafael Pastor. A branch and bound algorithm for the response time variability problem. Journal of Scheduling, 16(2):243–252, 2013.

[10] A. Garca-Villoria and S. Salhi. Scheduling commercial advertisements for television. International journal of production research, 53(4):1198–1215, Oct 2014.

[11] Alberto Garca-Villoria and Rafael Pastor. Solving the response time variability problem by means of a genetic algorithm. European Journal of Operational Research, 202(2):320–327, 2010.

[12] Alberto Garca-Villoria and Rafael Pastor. Minimising maximum response time. Computers & Operations Research, 40(10):2314–2321, 2013.

[13] Oktay Günlük, Jon Lee, and Janny Leung. A Polytope for a Product of Real Linear Functions in 0/1 Variables, pages 513–529. Springer New York, New York, NY, 2012.

[14] Ching-Chih Han, Kwei-Jay Lin, and Chao-Ju Hou. Distance-constrained scheduling and its applications to real-time systems. IEEE Trans. Computers, 45(7):814–826, 1996.

[15] R. Holte, A. Mok, L. Rosier, I. Tulchinsky, and D. Varvel. The pinwheel: A real-time scheduling problem. In 22nd Hawaii International Conference on System Sciences, Kailua-Kona, pages 693–702, 1989.

[16] Kenyonand and Schabanel. The data broadcast problem with non-uniform transmission times. Algorithmica, 35(2):146–175, 2003.

[17] Eun-Seok Kim and Celia A. Glass. Perfect periodic scheduling for three basic cycles. Journal of Scheduling, 17(1):47–65, 2014.

[18] Wieslaw Kubiak. Fair sequences. In In Handbook of Scheduling: Algorithms, Models and Performance Analysis, Leung, J.Y-T., editor, Chapman & Hall/CRC, Boca, 2004.

[19] J. Miltenberg. Level schedules for mixed-model assembly lines in just-in-time production systems. Management Science, 35(2):192–207, February 1989.

[20] Steven Morris. Interactive TV standards: a guide to MHP, OCAP, and JavaTV. Elsevier, San Diego, CA, 2005.

[21] C. A. Waldspurger and E. Weihl. W. Stride scheduling: Deterministic proportional- share resource management. Technical report, Cambridge, MA, USA, 1995.