





# Parking Scheduling Optimisation at Paris Charles de Gaulle International Airport

Thibault Falque<sup>1,2</sup>, Christophe Lecoutre<sup>2</sup>, Bertrand Mazure<sup>2</sup> and Romain Wallon<sup>2</sup>

<sup>1</sup>Exakis Nelite, France

<sup>2</sup>CRIL, Univ. Artois & CNRS, France

**Keywords:** Constraint Programming, Optimization, Planning, Application.

**Abstract:** Before the COVID-19 health crisis, the International Air Transport Association (IATA) forecasted that air passengers would almost double by 2036, reaching 7.8 billion people. More than ever, air transport players such as airline and airport companies, in a strongly competitive climate, need to benefit from a carefully optimized management of the airport resources in order to improve the quality of services and to control the induced costs. For example, the allocation of parking spaces for landing aircrafts remains a central issue at the airports, while optimizing an economic function determined by some business rules. In this paper, we investigate the Airport Parking Assignment Problem (APAP) with a Constraint Programming (CP) approach. We introduce a CP model, under the form of a Constraint Optimization Problem, and present some promising preliminary experimental results from data coming from ADP (Aéroports de Paris).

## 1 INTRODUCTION


Before the COVID-19 health crisis, the International Air Transport Association (IATA) forecasted that air passengers would almost double by 2036, reaching 7.8 billion people. In such a context, optimizing the management of the airport resources remains essential to control induced costs while keeping a good quality of services. For many planning and scheduling air transport problems, techniques and tools developed from mathematical and constraint programming remain essential. Specifically, when airline companies have access to the resources delivered at the airport, the consumption of these resources (e.g., check-in banks, aircraft parkings) must be carefully planned while optimizing an objective function determined by some business rules; see, for example, (Mangoubi and Mathaisel, 1985) (Lim et al., 2005) (Diepen et al., 2007). At airports, one of the significant combinatorial problems that need to be solved is the Stand Allocation Problem. This problem is closely related to the Gate Allocation Problem, and both have been extensively studied since the 1980s (Man-


goubi and Mathaisel, 1985) (Dincbas and Simonis, 1991) (Dorndorf et al., 2008) (Simonis, 2007). In the following, we will refer to both problems without distinction. The main objective of the Stand Allocation Problem is to find an optimal assignment of aircraft serving different flights to the available stands (*gates*) at the airport. Each flight requires a specific *stand* for various tasks, such as passenger boarding, baggage handling, and refueling.


**Definition 1** (Stand). A stand  $p \in S$  is an aircraft position. The first type of stands is contact stands (or hard stand) connected to a terminal by a door and a gateway while the second type requires a bus to reach the terminal (remote stands).


**Definition 2** (Flight turnaround or rotation). A flight turnaround (or rotation)  $\phi \in \Phi$  comprises at least one arrival or departure flight or both. We note  $a_\phi$  and  $d_\phi$  the rotation's start and end time, respectively (i.e., the arrival time of the flight at the airport and the departure time of the flight from the airport).

**Definition 3** (Stand operations). The stand operations of a flight turnaround can be divided into three parts: (i) operations about the arrival flight composed of the unboarding of passengers and luggages, (ii) waiting time, and (iii) operations of the departure flight composed of the boarding of passengers and luggages. Note that during these operations, we also have air-

<sup>a</sup> <https://orcid.org/0000-0003-2803-1530>

<sup>b</sup> <https://orcid.org/0000-0002-2205-6545>

<sup>c</sup> <https://orcid.org/0000-0002-3508-123X>

<sup>d</sup> <https://orcid.org/0000-0001-7200-4279>

craft ground handling operations: catering, refueling, cabin services, etc.

**Definition 4 (Operation).** An operation, denoted by  $t_i$ , is defined by its commencement at time  $a_i$  and its conclusion at time  $d_i$  at the designated stand for operation  $t_i$ . Each operation is associated with an aircraft of a specific model, such as an A320 or a B757. The aircraft type for operation  $t_i$  is identified by  $k_i$ . The aggregate of all flight operations is denoted as  $\mathcal{T}$ . For a specified rotation,  $\phi$ , its related set of activities, also known as the set of tasks, is represented as  $\mathcal{T}_\phi$ . This set comprises  $n$  individual tasks, expressed as  $\{t_i \mid i \in 1..n\}$ . We also introduce sets  $T_n$  which consist of tuples of tasks, each tuple containing  $n$  tasks.

Depending on the waiting time and for operational reasons, the aircraft may be moved to another stand, creating several flight operations. Figures 1 illustrate the possible cases. This movement requires a towing tractor and involves a cost for the operator. Figure 1a presents the case where the arrival and the departure flight are the same operations. Figures 1b and 1c show that the waiting time is enough to consider a decomposition on two or three operations, respectively.

In the context of airport operations, stand assignments are crucial and must be both aligned with the airport's services and convenient for passengers. According to (Dorndorf et al., 2007), these assignments are governed by stringent rules, such as ensuring that each stand is allocated to only one flight at a time, adhering to spatial limitations for adjacent stands, and considering the specific preferences of certain aircrafts for particular stand positions. The literature reveals a variety of objectives for addressing this problem. For example, studies have focused on minimizing passenger walking distances through methods like binary integer programming (Bihl, 1990) (Yan et al., 2002), or stochastic models (Yan and Tang, 2007). Another objective is to minimize the occurrence of off-gate events, as discussed in (Vanderstraeten and Bergeron, 1988). Moreover, multi-objective approaches have been explored:

- (Lim et al., 2005) applied integer programming to optimize both the reduction of delay penalties and the total walking distance.
- (Prem Kumar and Bierlaire, 2014) utilized binary integer programming for threefold optimization: maximizing the rest time between two turns at a gate, minimizing the towing cost for aircraft with extended turns, and reducing overall costs, including penalties for not assigning preferred gates to specific turns.
- Finally, (Dorndorf et al., 2012) employed a clique partitioning formulation to address four objectives: maximizing the total assignment prefer-

ence score, minimizing the number of unassigned flights and tows, and enhancing the robustness of the schedule.

The rest of this paper is organized as follows. In Section 2, we give some preliminaries regarding constraint programming and solving and the stand allocation problem as defined at Paris Airports. In Section 3, we present some modeling for the stand allocation problem. Before concluding, we present a few promising experimental results in Section 4.

## 2 PRELIMINARIES

### 2.1 Constraint Optimization Problem

CP (Constraint Programming) (Apt, 2003) (Rossi et al., 2006) (Lecoutre, 2009) is a powerful and recognized paradigm for modeling and solving everyday problems (for example, the timeschedule problem can be modeled using constraint programming), as well as combinatorial problems ranging from configuration and planning to bioinformatics. CP offers generic methods for modeling and solving this type of problems. It aims to reduce modeling complexity by being as close to the natural language description of the problem as possible. In the CP approach, users define the problem by specifying decision variables and constraints that define the relationships between these variables. The objective is to find an assignment for all variables that satisfies all the given constraints. This is known as a *Constraint Satisfaction Problem (CSP)*. The task is then solved by employing specialized solvers which use generic methods to efficiently explore the search space and find solutions that satisfy all the constraints. A *constraint network (CN)* is composed of a finite set of variables and a finite set of constraints. Each variable  $X$  takes its value in a finite set called *domain* of  $X$ , denoted  $\text{dom}(X)$ . Each constraint defines a relation on a set of variables. A *solution* of a CN is an assignment of values to all its variables such that all the constraints of the CN are satisfied. A CN is said to be *consistent* if it has at least one solution, and the corresponding decision problem, called *Constraint Satisfaction Problem (CSP)*, is to determine whether a CN is consistent. Deciding whether a CSP is satisfiable is an NP-complete problem (Mackworth, 1977). Other combinatorial tasks may interest: enumerating or counting the set of solutions, calculating an optimal solution according to a given objective, etc. We conclude this section by defining one such task: the constraint optimization problem. A *Constraint Optimization Problem (COP)* instance can be interpreted as a CSP instance with an associated

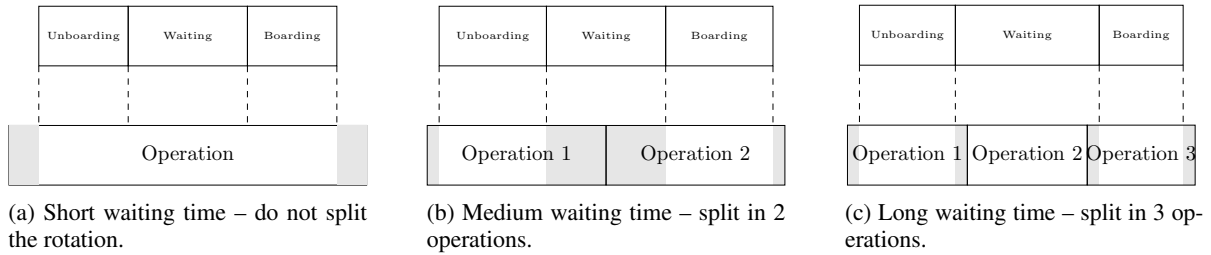


Figure 1: Number of operations depending on waiting time.

cost function. This function gives a numerical value to each solution of the instance, thereby quantifying its quality. The objective is to find the solution that maximizes or minimizes this function. For example, such a function may maximize (resp. minimize) the sum or product of certain variables (possibly associated with coefficients) of the problem.

## 2.2 Resolution Methods and Heuristics

Backtracking search is a conventional method for addressing COP instances, functioning as a complete procedure. It conducts a *depth-first exploration* of the search tree, facilitated by a backtracking mechanism alongside a sequence of decisions and propagations. There are also incomplete search techniques which do not ensure algorithmic completeness, but can still be more effective at locating solutions. Backtrack search for COP relies on CSP solving: the principle is to add a special *objective constraint*  $\text{obj} < \infty$  to the constraint network (although it is initially trivially satisfied), and to update the limit of this constraint whenever a new solution is found. It means that any time a solution  $S$  is found with cost  $B = \text{obj}(S)$ , the objective constraint becomes  $\text{obj} < B$ . Hence, a sequence of better and better solutions is generated (*satisfiability* is systematically proved with respect to the current limit of the objective constraint) until no more exists (*unsatisfiability* is eventually proved with respect to the limit imposed by the last found solution), guaranteeing that the last found solution is *optimal*. The order in which variables are chosen during the depth-first traversal of the search space is decided by a *variable ordering heuristic*. Each heuristic associates to a variable a score computed *statically*, *dynamically* or *adaptatively*. In this paper we focus on two heuristics. *dom/wdeg* is a classical heuristics (Boussemart et al., 2004) that aggregates by a division operator the *dom* heuristics with a dynamic degree of the variable *wdeg*. We also use a recent heuristics called *Frba/dom* (Li et al., 2021) based on the *fail first principle* and exploiting two aspects of failure information collected during the search: the failure proportion after the propagations of assignments

of variables and the failure length heuristics considering the length of failures, which is the number of fixed variables composing a failure. Similar to variable selection heuristics, a *value ordering heuristic* is essential to determine the subsequent value to assign. A straightforward heuristic approach involves utilizing the initial value in the domain, symbolized by *First*. This method is frequently adopted due to its robustness. Recently, a heuristic strategy suggesting the use of the value having the most significant effect on the objective function was introduced in (Fages and Prud'Homme, 2017), labeled as *Bivs*. For COPs, it is often advantageous to prioritize the value observed in the most recent solution using *solution saving*, as mentioned in works like (Vion and Piechowiak, 2017) (Demirovic et al., 2018).

## 2.3 Stand Allocation Formulation

In this section, we consider the formulation of the Gate Allocation Problem proposed by (Dorndorf et al., 2008) and (Guépet et al., 2015) which we will adapt for a Stand Allocation Problem (without any loss of generality) and to the case of Paris Airports. The previous section explains that the rotation can be decomposed in several operations. At Paris Airports, there can only be two movements for the same aircraft, i.e., a maximum of 3 parking positions, and the conditions for determining whether a rotation must be decomposed depend on certain processing times. Some physical constraints exist, as imposed by the airport infrastructure.

**Rule 1 (Capacity).** *The capacity constraints prevent certain aircraft types from being placed on some parking.*

**Rule 2 (No-Overlap).** *The No-Overlap constraints reflect the physical impossibility of assigning two operations (two flights) to the same parking. An operation  $t_i \in \mathcal{T}$  overlaps with another operation  $t_j \in \mathcal{T}$  if  $a_i < d_j \wedge a_{t_j} < d_i$ . The set of operations overlapping with  $t_i$  is denoted by  $O_i$ , and so, contains all operations  $t_j$  overlapping with  $i$ .*

**Rule 3 (Shading constraints).** *The shading constraints block the positioning of an aircraft*

on some nearby parking (regardless of the type of aircrafts, e.g., two aircrafts cannot be simultaneously assigned to adjacent stands due to space limitations).

**Example 1** (Shading constraints). For example, Figure 2 shows that, if an aircraft is placed on parking A14 then the parking A16 is “shaded” and vice-versa (it is symmetrical).

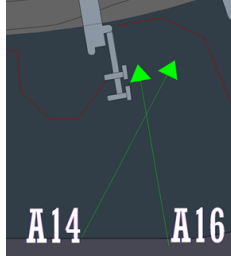


Figure 2: Example for the shading constraint.

**Rule 4** (Reduction). The reduction constraints are similar to the shading constraints except that they consider aircraft types. These constraints can be defined by 4-tuples  $(k, p, k', \Delta)$  with  $\Delta \subset S$  being a set of parkings. For every parking  $p'$  in  $\Delta$ , a plane of type  $k'$  is allowed on  $p'$  if a plane of type  $k$  is placed on  $p$ . We note  $\mathcal{D}$  the set of all such reductions (all 4-tuples).

**Example 2** (Reduction constraint). As an illustration, considering Figure 3, shading will only be effective if a specific aircraft type has been placed on A10, then the reduction will still allow a subset of aircraft types to be placed on A8 and A12.

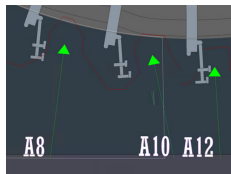


Figure 3: Example for the reduction constraint.

**Rule 5** (Order). The order constraints impose for two parkings  $p_1, p_2$  that the aircraft put on  $p_1$  must arrive and leave before that put on  $p_2$ , if the two aircrafts have an overlapping time.

**Example 3** (Order constraint). Terminal 1 of CDG (Charles de Gaulle) has a specific infrastructure that imposes an order constraint. Let us take a look at Figure 4: if a plane is placed on Y07 and another plane is placed on Y06, we need to make sure that the plane on Y07 leaves before the plane on Y06, because planes arrive on the outside (route marked by A) and leave on the inside (route A3). Note that the direction is reversed if we take stands Z02 and Z01.

We note  $OR_{\mathcal{R}}$  the set of pair of parking  $p_1, p_2$  while  $p_1$  is before  $p_2$ .

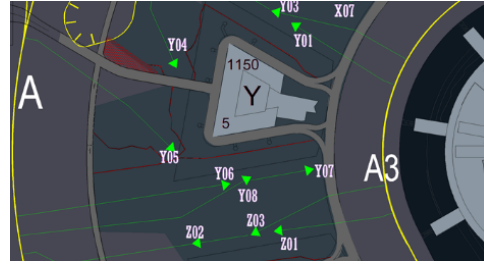


Figure 4: Example of a special traffic situation linked to the CDG1 infrastructure.

Frequently, some stands are unavailable for several hours to several days (for example, for maintenance reasons).

**Rule 6** (Unavailable constraints). The unavailable constraints ensure that certain stands are not available for a period of time (which may be periodic). Said differently, we must remove from the domain the stand for each operation that overlaps with the period of exclusion. We note  $u = (p, s, e)$  a triplet where  $p$  is the stand to exclude,  $s$  and  $e$  is the start and end time of the excluded period.

Another type of unavailability, specific to certain rotations, consists in declaring a list of prohibited stands for a given rotation.

**Rule 7** (Exclusion). The exclusion constraints ensures that certain stands are excluded from certain rotations under some conditions.

The components for the stand allocation problem formulation at Paris Airports are given below:

- $\mathcal{T}$  the set of operations. The previous section explains that an operation  $t_i \in \mathcal{T}$  is defined by a start time  $a_i$  and an end time  $d_i$ . For each operation  $t_i$ , we also have the set of operations that overlap with  $t_i$ :  $O_{t_i}$ .
- $S$  the set of the stands.
- $S_i \subset S$  the set of compatible stands (stands with a capacity compatible with  $t_i$ ) for operation  $t_i$ .
- $OR_{\mathcal{R}}$  the set of pair of parkings for order rules.
- $\mathcal{U}$  the set of unavailable rules (i.e., the set of all triplet  $(p, s, e)$ ).
- $\mathcal{E}_i$  the set of excluded stands for an operation  $t_i$ .
- $Q \subseteq S^2$  the set of shadow restrictions. If  $(p_1, p_2) \in Q$  then two operations that overlap cannot be placed at the same time on  $p_1$  and  $p_2$ .
- $\mathcal{D}$  the set of reductions. For a quadruplet  $(k, p, k', \Delta) \in \mathcal{D}$  and an operation  $t_i \in \mathcal{T}$  with an aircraft type  $k$ , if the operation  $t_i$  is assigned to the parking  $p$ , then only operations with an aircraft type  $k'$  are allowed to parkings of the set  $\Delta$ .

- $\mathcal{M}\mathcal{P} = (m_{i,p})_{\mathcal{T} \times \mathcal{S}}$  the affinity matrix, i.e.,  $m_{i,p}$  is the airline satisfaction realized if operation  $t_i \in \mathcal{T}$  is assigned to stand  $p$ .

An assignment  $I$  is a mapping between operations  $\mathcal{T}$  and stands  $\mathcal{S}$ . The quality  $\text{Obj}(I)$  of an assignment  $I$  is defined by  $\text{Obj}(I) = \mathcal{C}(I)$  where  $\mathcal{C}$  is the total operation-stand affinity. Our objective is to find an assignment maximizing  $\text{Obj}(I)$  while respecting operation-stand compatibilities, shadowing, reduction restrictions and overlapping constraints.

### 3 MODELING

For modeling COP, we have chosen to use the recently developed Python library `PyCSP3` (Lecoutre and Szczepanski, 2020) that permits to generate specific instances (after providing ad hoc data) in `XCSP3` (Boussemart et al., 2020), which is recognized by CP solvers such as ACE (AbsCon Essence) (Lecoutre, 2023) and `Choco` (Prud'homme et al., 2016). In the following, we show how to model the stand allocation problem in the context of Paris Airports.

First, we need to introduce the variables of our model. Actually, in addition to a stand-alone variable used to count the number of rotations that are not splitted, we need two arrays of variables to represent assigned stands and their associated rewards:

- $x$  is a matrix of  $|\mathcal{T}|$  variables having the set of values  $\{0, \dots, |\mathcal{S}| - 1\}$  as domain;  $x_i$  represents the index (code) of the stand assigned to the task  $t_i$ .
- $r$  is a matrix of  $|\mathcal{T}|$  variables having the set of values  $\{0, \dots, 100\}$  as domain;  $r_i$  represents the satisfaction of the company for task  $t_i$ .

We now introduce constraints for this problem. An illustration is now given to facilitate the understanding of the various constraints of the model.

**Example 4.** Let us consider an example with a set of 5 rotations  $\{\phi_1, \dots, \phi_5\}$ , and a set of 4 stands  $\{p_1, \dots, p_4\}$ . where  $p_3$  and  $p_4$  are assumed to be remote stands; so, we have  $S_{\pi} = \{p_4, p_5\}$ . Information concerning rotations is given in Table 1.

In the next sections we introduce different modeling for the problem. Given the nature of the problem (and data), it is natural to post so-called *extensional constraints*, which explicitly enumerate either the *allowed tuples* (positive table) or the *disallowed tuples* (negative table) for a sequence of variables (representing the scope of a constraint). Over the last decade, efficient algorithms have been developed to handle such table constraints (Lecoutre, 2011) (Lecoutre et al., 2015) (Demeulenaere et al., 2016) (Verhaeghe et al., 2017).

#### Classical Variant

**Model 1** (Classical variant).

$$x_j \in S_{\text{remote}}, \forall (t_i, t_j, t_k) \in \mathcal{T}_3 \quad (C_1)$$

$$\langle x_j \rangle \in S_i, \quad (C_2)$$

$$\langle x_i, x_j \rangle \notin \{(p_1, p_2), \forall p_1, p_2 \in \mathcal{Q}\} \\ \forall t_i \in \mathcal{T}, \forall t_j \in O_i \quad (C_3)$$

$$\langle x_i, x_j \rangle \notin \{(p_1, p_2) \mid p_2 \in \delta\}, \forall t_i \in \mathcal{T}, \\ \forall \langle k, p_1, k', \delta \rangle \in \mathcal{D} \mid k = \text{kind}(i), \\ \forall j \in O_i \mid k' \neq \text{kind}(j) \quad (C_4)$$

$$\langle x_j, r_j \rangle \in \{(p_1, r_j^1) \mid p_1 \in S_i\}, \forall t_i \in \mathcal{T} \quad (C_5)$$

$$\langle x_i, x_j \rangle \notin O\mathcal{R}, \forall t_i \in \mathcal{T}, \forall j \in O_i \quad (C_6)$$

$$x_i \neq p, \forall t_i \in \mathcal{T}, \quad (C_7)$$

$$\forall (p, s, e) \in \mathcal{U} \mid \text{overlap}(t_i, s, e) \quad (C_8)$$

$$x_i \notin s, \forall i \in \mathcal{T}, \forall s \in \mathcal{E}_i \quad (C_8)$$

Note that Constraint  $C_1$  forces the middle parking to be remote.

**Example 5.** For our example:  $\mathcal{T}_1 = \{\phi_1\}$ ,  $\mathcal{T}_2 = \{\phi_2, \phi_3\}$ ,  $\mathcal{T}_3 = \{\phi_4, \phi_5\}$ . The set of operations  $\mathcal{T}$  is composed of each tasks from each rotation:  $\{t_1, \dots, t_{11}\}$  where  $t_1$  corresponds to the only task of  $\phi_1$  while  $t_{11}$  corresponds to the third tasks of  $\phi_5$ . So we must post the constraints:  $x_7 \in S_{\text{remote}}$  and  $x_{10} \in S_{\text{remote}}$ .

To enforce capacity rules, we post unary constraints (see Constraint  $C_2$ ).

**Example 6.** Table 2 provides each parking capacity for our example. Based on this table, we must post the following unary constraints: For  $\phi_1$ :  $x_1 \in \{p_1, p_3, p_4, p_5\}$ . For  $\phi_2$ :  $x_2 \in \{p_1, p_2, p_3, p_4, p_5\}, x_3 \in \{p_1, p_2, p_3, p_4, p_5\}$ . For  $\phi_3$ :  $x_4 \in \{p_1, p_2, p_3, p_4, p_5\}, x_5 \in \{p_1, p_2, p_3, p_4, p_5\}$ . For  $\phi_4$ :  $x_6 \in \{p_3, p_4, p_5\}, x_7 \in \{p_3, p_4, p_5\}, x_8 \in \{p_3, p_4, p_5\}$ . For  $\phi_5$ :  $x_9 \in \{p_4, p_5\}, x_{10} \in \{p_4, p_5\}, x_{11} \in \{p_4, p_5\}$ .

Let us recall that when a parking  $p_1$  is shaded by a parking  $p_2$  then these two values cannot be assigned together to any pair of overlapping tasks. This leads to binary negative table constraints. Although not explicitly shown below, assigning the same value twice for any pair of overlapping operations is also forbidden (see Constraint  $C_3$ ). Constraint  $C_4$  represents the reduction constraint and is defined with binary negative tables.

Table 1: Data about rotations.

Rot.	airline	ntasks	kind	$a_\phi$	$d_\phi$
$\phi_1$	$a_1$	1	$k_1$	8h	10h
$\phi_2$	$a_2$	2	$k_2$	8h	12h
$\phi_3$	$a_2$	2	$k_2$	12h	16h
$\phi_4$	$a_3$	3	$k_3$	9h	15h
$\phi_5$	$a_3$	3	$k_4$	12h	18h

**Example 7.** For our example, suppose that a reduction constraint exists between the stands  $p_1$  and  $p_2$ , given by  $\mathcal{D} = \{(k_1, p_1, k_3, \{p_2\}), (k_2, p_1, k_4, \{p_2\})\}$ . We also have  $O_1 = \{(t_2), (t_6)\}$  and  $S_{\phi_1} = \{p_1, p_3, p_4, p_5\}$ . Task 1 from  $\phi_1$  overlaps Task 1 of  $\phi_2$ . Recall that  $\text{kind}(\phi_2) = k_2$ . So, for the reduction imposed by  $\phi_1$ , we add the constraint that forbid the pair of values  $(p_1, p_2)$  for the pair of variables  $\langle x_1, x_2 \rangle$ . In other words, it is forbidden to use  $p_2$  with the rotation  $\phi_2$  after placing  $\phi_1$  on  $p_1$  (its capacity is reduced), i.e.,  $\langle x_1, x_2 \rangle \notin \{(p_1, p_2)\}$ . Although  $\phi_1$  overlaps with  $\phi_4$ , there are no restrictions to consider with  $\phi_1$  as  $\phi_4$  has a capacity of type  $k_3$  (see Table 1) which is allowed in relation to the reduction.

According to the airlines preferences from affinity matrix  $\mathcal{MP}$ , we can post binary table constraints to “compute” rewards when filtering such constraints (see Constraint  $C_5$ ).

Table 3: Affinity matrix.

Airline / Parking	$p_1$	$p_2$	$p_3$	$p_4$	$p_5$
$a_1$	75	75	100	50	50
$a_2$	60	0	100	50	50
$a_3$	0	100	80	50	50

**Example 8.** For our example, let us assume that rewards are given by the matrix in Table 3. From the data in this table, we post the following constraint for the first rotation (the same principle is adopted for the other rotations):

$$\langle x_1, r_1 \rangle \in \{(p_1, 75), (p_2, 75), (p_3, 100), (p_4, 50), (p_5, 50)\},$$

Considering the reward variables we can express the objective function as follows:

$$\text{maximize } \sum_{t_j \in \mathcal{T}} w_j \cdot r_j$$

This function aims to optimize the cumulative weighted rewards across all tasks within each rotation. In this context,  $w_j$  is the weight attributed to the task  $t_j$ . This weight quantifies the importance or priority of each task. The term  $r_j$  represents the decision

Table 2: Capacity.

Pkg	Capacity
$p_1$	$\{k_1, k_2\}$
$p_2$	$\{k_2, k_3\}$
$p_3$	$\{k_1, k_2, k_3\}$
$p_4$	$\{k_1, k_2, k_3, k_4\}$
$p_5$	$\{k_1, k_2, k_3, k_4\}$

variables from the matrix  $r$ , specifically the weight allocated to the resources assigned for task  $t_j$ . Thus, the objective function is to maximize the total value by strategically allocating resources to tasks based on their importance and the associated rewards.

### AllDifferent Variant

In this variant, we propose to modify the shading constraint by posting AllDifferent constraints in addition to the table constraints already proposed, as it is often done in the state-of-the-art (Simonis, 2007) (Dincbas and Simonis, 1991). In these approaches, an AllDifferent constraint is added between all overlapping pairs of tasks, but this cannot work here, as it would be less restrictive than the shadow constraint. Indeed, an AllDifferent constraint would force two operations  $t_i$  and  $t_j$  to be assigned different parkings. However, by assigning  $p_1$  and  $p_2$  as parkings for these tasks, the constraint would be respected but would violate the shadow constraint if  $p_1$  shaded  $p_2$ .

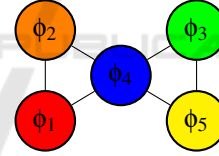


Figure 5: Interval graph based on data from Table 1.

Nevertheless, it is possible to use AllDifferent constraints with an interval graph. Each vertex of the graph represents a task (the task’s time interval) and is connected by an edge to another vertex if and only if there is an overlap between the two intervals. Figure 5 represents an interval graph  $\mathcal{G}$  for our example, each vertex represents an interval and there is an edge between intervals when they intersect. From this graph, we need to post an AllDifferent constraint for each maximum clique in the graph.

**Definition 5 (Maximum clique).** A maximum clique of  $\mathcal{G}$  has the greatest number of vertices, which is maximal for the cardinal. We note  $C_{\mathcal{G}}$  the set of all maximum cliques.

For the set of maximum cliques, we can post the constraints:  $\text{allDifferent}(\{x_i, \forall t_i \in c\}), \forall c \in C_{\mathcal{G}}$ .

For our example and based on our interval graph (Figure 5), the maximum cliques are:  $\{\phi_1, \phi_2, \phi_4\}$  and

Table 4: General information about the parking planning.

Airport	Terminals	Week	#Rot
CDG	T2B T2D	WE29	755
CDG	T1 T3	WE29	888
CDG	T2B T2D	WE30	763
CDG	T1 T3	WE30	903
CDG	T1 T3	WE34	941
CDG	T1 T3	WE35	960
CDG	T2B T2D	WE36	785
CDG	T1 T3	WE36	884
CDG	T2B T2D	WE37	757
CDG	T1 T3	WE37	868
CDG	T2B T2D	WE38	765
CDG	T1	WE38	643

$\{\phi_3, \phi_4, \phi_5\}$ .

## 4 EXPERIMENTAL RESULTS

This section presents some experimental results of our modeling presented in the previous sections. In the context of our experiments, we used instances in XCSP format (Boussemart et al., 2020) generated with PyCSP<sup>3</sup> (Lecoutre and Szczechanski, 2020). Table 4 presents some factual information about the different plannings used for these experiments. The first two columns indicate the area of the planning (i.e., Airport and Terminals concerning the planning). The third column gives the date of the planning. Finally, the last column displays the number of rotations. For each planning, we have considered the two variants of the problem: `classical` which contains the constraints given in Modeling 1 and `alldiff` in which we add the `allDifferent` constraint based on the maximum cliques (see Section 3).

We use different configurations of the solver ACE as presented below and the current solution used by ADP for planning resources. We call this approach ADP. ACE-based solvers are named as follows:

$$\left\{ \begin{array}{l} f \in \{\text{classical}, \text{alldiff}, \text{notbreak}\} \\ \text{varh} \in \{\text{Frba}/\text{dom}, \text{Wdeg}\} \\ \text{valh} \in \{\text{first}, \text{Bivs}\} \end{array} \right\}$$

All solvers have been run on a cluster of computers equipped with 128 GB of RAM and two quadcore Intel XEON E5-2637 (3.5 GHz). The time was limited to 5 minutes and the memory to 64 GB of RAM.

Figure 6 displays the evolution of solver-determined bounds for some instances, chosen due to space constraints<sup>1</sup>. This figure uses the x-axis to represent the elapsed time and the y-axis to indicate the

<sup>1</sup>All the experiments data will be made publicly available if the paper is accepted

value of the bound. The blue dashed line marks the bounds set by the previous ADP system, which had an average response time of 7.5 seconds. This system adopted a strategy of accepting the first solution and then applying localized optimization. On all tested instances, our approach consistently outperformed the ADP system in terms of achieving superior bounds. It is important to note, however, that the attainment of optimal bounds might occur later in our process. The figure also shows that the `AllDifferent`-based method, labeled as `allDiff+extension`, closely parallels the traditional approach. It should be mentioned that the time taken to convert each instance into XML format is not reflected in the timelines shown in this figure. The exclusion of this compilation time is due to its inclusion of data processing durations, which are expected to be absent in the final production version. In the production environment, this processing will be conducted prior to utilizing the PyCSP<sup>3</sup> compiler and will employ a faster programming language.

## 5 CONCLUSION AND PERSPECTIVES

In this paper, various models – mainly exploiting table constraints – addressing the stand allocation problem have been introduced. This problem is a fundamental aspect of airport operations that has implications for both efficiency and airline satisfaction. We put forward two distinct formulations of the problem: the `classical` variant which mainly uses extensional constraints and the `allDifferent` variant which adds `allDifferent` constraints for each maximum clique of the interval graph formed from overlapping tasks. Subsequently, we conducted an analysis of different configurations for the ACE solver and compared the results with the current ADP method. The value selection heuristic `Bivs` shows very good performance and always obtains a better bound than that found by

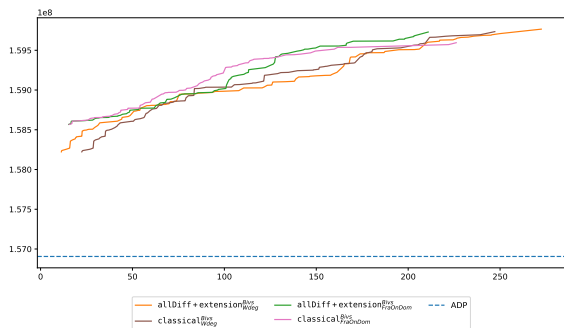


Figure 6: Evolution of the bound for instance of CDG T1 T3 and planning for week 29.

ADP. An important aspect of our current approach is its foundation on open-source tools, which presents a significant advantage over the former solution that was commercial. This transition not only offers potential cost savings but also enhances adaptability and accessibility of the tools, as desired by the direction of Paris Airports. In the future, we plan to make further experiments and to study the interest of using multi-objective constraint solvers like Choco.

## REFERENCES

- Apt, K. R. (2003). *Principles of Constraint Programming*. Cambridge University Press.
- Bihl, R. A. (1990). A conceptual solution to the aircraft gate assignment problem using 0, 1 linear programming. *Computers & Industrial Engineering*, 19(1-4):280–284.
- Boussemart, F., Hemery, F., Lecoutre, C., and Sais, L. (2004). Boosting systematic search by weighting constraints. pages 146–150.
- Boussemart, F., Lecoutre, C., Audemard, G., and Piette, C. (2020). XCSP3-core: A format for representing constraint Satisfaction/Optimization problems. *CoRR*, abs/2009.00514.
- Demeulenaere, J., Hartert, R., Lecoutre, C., Perez, G., Peron, L., Régis, J.-C., and Schaus, P. (2016). Compact-Table: Efficiently Filtering Table Constraints with Reversible Sparse Bit-Sets. In *Proceedings of CP'16*, pages 207–223.
- Demirovic, E., Chu, G., and Stuckey, P. (2018). Solution-based phase saving for CP: A value-selection heuristic to simulate local search behavior in complete solvers. In *Proceedings of CP'18*, pages 99–108.
- Diepen, G., Akker, J., Hoogeveen, J., and Smeltink, J. (2007). Using column generation for gate planning at Amsterdam Airport Schiphol.
- Dincbas, M. and Simonis, H. (1991). APACHE - A constraint based, automated stand allocation system. Automated Stand Allocation System Proc. Of Advanced Software Technology in Air Transport (ASTAIR'91) Royal Aeronautical Society, pages 267–282.
- Dorndorf, U., Drexler, A., Nikulin, Y., and Pesch, E. (2007). Flight gate scheduling: State-of-the-art and recent developments. *Omega*, 35(3):326–334.
- Dorndorf, U., Jaehn, F., and Pesch, E. (2008). Modelling Robust Flight-Gate Scheduling as a Clique Partitioning Problem. *Transportation Science*, 42(3):292–301.
- Dorndorf, U., Jaehn, F., and Pesch, E. (2012). Flight gate scheduling with respect to a reference schedule. *Annals of Operations Research*, 194(1):177–187.
- Fages, J.-G. and Prud'Homme, C. (2017). Making the First Solution Good! In *ICTAI 2017*, pages 1073–1077, Boston, MA. IEEE.
- Guépet, J., Acuna-Agost, R., Briant, O., and Gayon, J. (2015). Exact and heuristic approaches to the airport stand allocation problem. *European Journal of Operational Research*, 246(2):597–608.
- Lecoutre, C. (2009). *Constraint Networks: Techniques and Algorithms*. ISTE/Wiley.
- Lecoutre, C. (2011). STR2: Optimized Simple Tabular Reduction for Table Constraints. *Constraints : an international journal*, 16(4):341–371.
- Lecoutre, C. (2023). ACE, a generic constraint solver. *CoRR*, abs/2302.05405.
- Lecoutre, C., Likitvatanavong, C., and Yap, R. (2015). STR3: A path-optimal filtering algorithm for table constraints. *Artificial Intelligence*, 220:1–27.
- Lecoutre, C. and Szczepanski, N. (2020). PyCSP3: Modeling combinatorial constrained problems in Python. *CoRR*, abs/2009.00326.
- Li, H., Yin, M., and Li, Z. (2021). Failure Based Variable Ordering Heuristics for Solving CSPs. In Michel, L. D., editor, *CP 21*, volume 210, pages 9:1–9:10.
- Lim, A., Rodrigues, B., and Zhu, Y. (2005). Airport Gate Scheduling with Time Windows. *Artificial intelligence review*, 24(1):5–31.
- Mackworth, A. K. (1977). Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118.
- Mangoubi, R. S. and Mathaisel, D. F. X. (1985). Optimizing Gate Assignments at Airport Terminals. *Transportation Science*, 19(2):173–188.
- Prem Kumar, V. and Bierlaire, M. (2014). Multi-objective airport gate assignment problem in planning and operations. *Journal of Advanced Transportation*, 48(7):902–926.
- Prud'homme, C., Fages, J.-G., and Lorca, X. (2016). Choco-solver, TASC, INRIA Rennes, LINA, Cosling S.A.
- Rossi, F., van Beek, P., and Walsh, T., editors (2006). *Handbook of Constraint Programming*. Elsevier.
- Simonis, H. (2007). Models for global constraint applications. *Constraints : an international journal*, 12(1):63–92.
- Vanderstraeten, G. and Bergeron, M. (1988). Automatic assignment of aircraft to gates at a terminal. *Computers & Industrial Engineering*, 14(1):15–25.
- Verhaeghe, H., Lecoutre, C., and Schaus, P. (2017). Extending Compact-Table to Negative and Short Tables. In *Proceedings of AAAI'17*, pages 3951–3957.
- Vion, J. and Piechowiak, S. (2017). Une simple heuristique pour rapprocher DFS et LNS pour les COP. In *Proceedings of JFPC'17*, pages 39–45.
- Yan, S., Shieh, C.-Y., and Chen, M. (2002). A simulation framework for evaluating airport gate assignments. *Transportation Research Part A: Policy and Practice*, 36(10):885–898.
- Yan, S. and Tang, C.-H. (2007). A heuristic approach for airport gate assignments for stochastic flight delays. *European Journal of Operational Research*, 180(2):547–567.