Exact and heuristic approaches to the airport stand allocation problem

Julien Guépet, Rodrigo Acuna Agost, Olivier Briant, Jean-Philippe Gayon

Exact and Heuristic Approaches to the Airport Stand Allocation Problem

J. Guépet[a,b], R. Acuna-Agost[b], O. Briant[a], J.P. Gayon[a]

[a]*Grenoble-INP / UJF-Grenoble 1 / CNRS, G-SCOP UMR5272 Grenoble, F-38031, France*
[b]*Amadeus S.A.S., 485 Route du Pin Montard, 06560 Sophia Antipolis, France*

Abstract

The Stand Allocation Problem (SAP) consists in assigning aircraft activities (arrival, departure and intermediate parking) to aircraft stands (parking positions) with the objective of maximizing the number of passengers/aircraft at contact stands and minimizing the number of towing movements, while respecting a set of operational and commercial requirements. We first prove that the problem of assigning each operation to a compatible stand is NP-complete by a reduction from the circular arc graph coloring problem. As a corollary, this implies that the SAP is NP-hard. We then formulate the SAP as a Mixed Integer Program (MIP) and strengthen the formulation in several ways. Additionally, we introduce two heuristic algorithms based on a spatial and time decomposition leading to smaller MIPs. The methods are tested on realistic scenarios based on actual data from two major European airports. We compare the performance and the quality of the solutions with state-of-the-art algorithms. The results show that our MIP-based methods provide significant improvements to the solutions outlined in previously published approaches. Moreover, their low computation make them very practical.

Keywords: Mixed integer programming, gate assignment problem, heuristic algorithms

1. Introduction

Every day, airports deal with different decisions related to aircraft movements. These decisions usually involve the use of fixed and limited resources such as runways, stands (parking positions) and passenger gates. Due to the growing flow of passengers, these resources are falling short of needs while activity planning is increasingly crucial and complex. Consequently, some airports have experienced deterioration in service quality. In one of our partner airports, the number of passengers allocated to remote stands has increased in the last years. This affects passenger connection times, increases bus transfer costs and decreases airport revenue given that airlines usually pay lower fees for flights allocated to remote stands. Since building new terminal gates is expensive and does not provide a short-term solution, value can only be gained from better management of airport resources.

In this paper we deal with the Stand Allocation Problem (SAP). This consists in assigning aircraft operations to available stands in line with operational requirements and different objectives. This problem is closely related to the Gate Allocation Problem (GAP). Our work results from close collaboration between the laboratory G-Scop and the company Amadeus. In what follows, we provide a detailed description of the stands, aircraft operations, operational requirements and the different objectives to be taken into account for solving the SAP.

A stand is an aircraft parking position. Figure 1 illustrates the two types of possible stands: contact stands (i.e., stands touching an airport terminal gate) and remote stands (i.e., stands where a bus is needed to reach the terminal). Airports and airlines usually prefer contact stands as they are more convenient for passengers and no bus transfer is necessary.

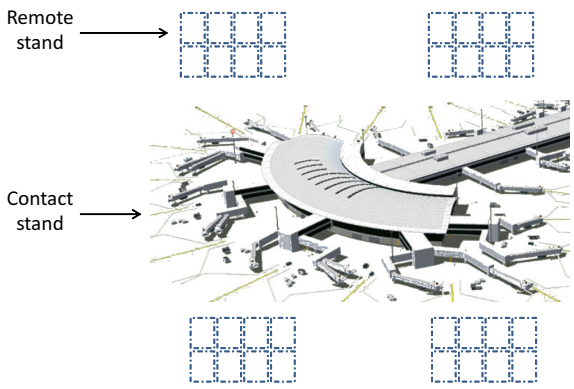
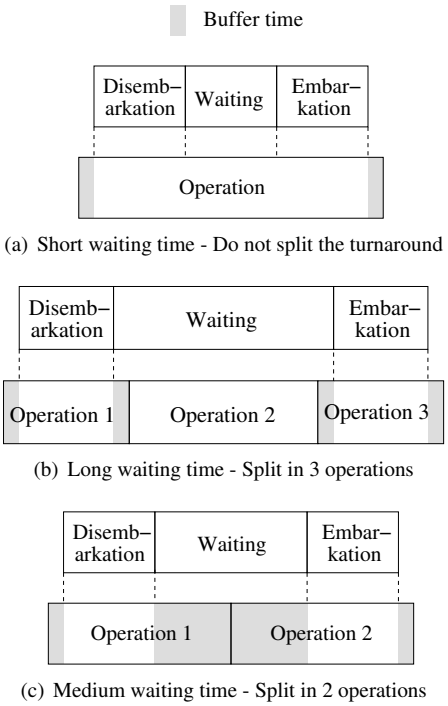


Figure 1: Airport stands

The stand operations of an aircraft turnaround can be roughly divided into three parts: disembarkation of the arrival flight, waiting, and embarkation of the departure flight. Disembarkation concerns passengers and luggage and also involves aircraft ground handling operations (refueling, cabin services, catering, etc.) linked to the aircraft's arrival. Similarly, embarkation concerns passengers and luggage and other related ground handling operations. The waiting period can be null if the turnaround is short. During the waiting period, airport operators may decide to tow (move) aircraft to other stands. This can be for several reasons but usually targets a better utilization of valuable stands (e.g. contact stands). However, these operations require an expensive towing tractor (see Figure 2) and increase airport congestion. The data provided by our partner airports shows that, at most, two towing operations are performed during a turnaround: one after disembarkation and one before embarkation. Consequently, we assume that turnarounds are split into three operations at most.

Figure 2: A towing tractor

In order to define operations, we need to distinguish between three situations depending on the waiting period length (see Figure 3). If the waiting period is too short to move the aircraft (case (a)), then we consider that we only have to schedule a single operation since disembarkation, waiting and embarkation will necessarily take place at the same stand. In order to make the assignment plan robust in the face of small disruptions such as short delays or early arrivals, we add a buffer time at the beginning and end of this single operation. If the waiting period is long enough to move the aircraft twice (case (b)), then we split the turnaround into three operations since an aircraft can potentially disembark at one stand, wait at a second stand and embark at a third stand. We add a buffer time before and after embarkation and disembarkation operations. If the duration of the waiting period is only long enough to move the aircraft once but not twice (case (c)), then the turnaround is split into two operations with the waiting time equally distributed between both operations and providing of a buffer time. Note that a different distribution of the waiting time is possible, but the one described above seems to be the most natural. We also add a buffer time before the embarkation operation and after the disembarkation operation. When towing is allowed (cases (b) and (c)), the towing time is much shorter than the disembarkation and embarkation times. Hence these can be included in the operations, which simplifies modeling even if it results in a slight overestimation of processing times. Indeed, this approach gives flexibility for actually performing the towing during the operations. In what follows, the set of operations, with fixed start and end time, is considered an input of the problem and is given by the airport.



(a) Short waiting time - Do not split the turnaround

(b) Long waiting time - Split in 3 operations

(c) Medium waiting time - Split in 2 operations

Figure 3: Splitting turnarounds in operations and adding buffer times

The assignment of aircraft operations to stands must take into account aircraft-stand compatibility. Indeed, not all aircraft can be assigned to all the stands because of size compatibility but also because of aircraft flight requirements. For example, some stands are forbidden to international flights because they do not offer access to governmental inspection facilities. Furthermore, two overlapping operations must not be assigned to the same

stand. Finally, adjacency conflicts, also called shadow restrictions, must be taken into account, e.g. two large aircraft cannot be assigned to adjacent stands simultaneously.

The quality of an assignment plan can be defined using several, often competing criteria, such as the number of unassigned operations, the number of passengers at contact stands, compliance with airline preferences, passenger connection convenience or the number of towing operations. In practice, an unassigned operation has to be handled manually, either overstepping certain requirements or delaying a flight. One option is to assign an operation to a non compatible stand and to transfer passengers to a compatible terminal area by bus. Another option is to keep the aircraft waiting on the tarmac.

In the literature, several authors consider the objective of minimizing passengers' walking distance or connection time (see Section 2). However, this is not always a suitable approach for airports since a large share of their revenue comes from the shops hosted in the terminal. The more passengers walk, the more likely they are to go into a shop and buy something thus boosting the airport's revenue.

For our partner airports, the assignment of aircraft activities is generally decided, at the latest, the day before the operations. In this phase, computation time is not overly problematic. However, on the day the operations are scheduled, disruptions can happen. Many random events may occur, leading to delays and flight cancellations. New flights (e.g. general aviation) and diversions can also impact planning. Hence, the assignment must be robust in the sense that small disruptions must not oblige airport authorities to change the whole assignment plan. Bigger disruption may oblige the airport to reassign aircraft. In this case, computation times need to be very short.

The Stand Allocation Problem (SAP) is closely related to the Gate Allocation Problem (GAP). A gate is the boarding desk where passengers' tickets are checked by the airline and a stand is the position where the aircraft is parked. In many US airports, embarking and disembarking passengers at remote stands is forbidden. Consequently, there is a perfect match between stands and gates, and therefore between the SAP and the GAP. In Europe, this is not often the case since embarking and disembarking can be done at a remote stand that can be associated with different gates (called bus gates). As we work with European airports, we will use SAP terminology.

To explore the SAP, this paper has been organized in several sections. A review of the literature and a summary of our contributions are presented in Section 2. In Section 3, we formally introduce the SAP and associated feasibility problem. Section 4 proves the NP-hardness of SAP and the NP-completeness of the associated feasibility problem. Section 5 presents a mixed integer programming formulation and a number of improvements designed to strengthen. Section 6 presents two MIP based heuristic algorithms. Computational experiments are presented in Section 7 to show the efficiency of the model and the performance of heuristic algorithms for realistic instances. The conclusion and a discussion of future prospects are finally given in Section 8.

2. Literature review

This literature review focuses on deterministic approaches related to mathematical programming for the GAP. More references to stochastic and expert system approaches can be found in the survey by Dorndorf et al. (2007).

The GAP has been widely studied since 1980. Many models aim at minimizing passenger walking distances or connection times, which naturally leads to a 0-1 quadratic integer program (QIP) close to the Quadratic Assignment Problem. Different methods can be found for solving it. Mangoubi and Mathaisel (1985) propose using the average distance of a gate to other gates and a greedy algorithm to solve the integer program (IP) thus obtained. Yan and Chang (1998) use the same assumption for modeling walking distance and propose a multi-commodity network flow model. They propose a Lagrangian relaxation solved by a sub-gradient algorithm and heuristics. Haghani and Chen (1998) use the classical linearization of the product of binary variables and propose a heuristic algorithm that consists in iterating a greedy algorithm. Xu and Bailey (2001) consider the same model and solve it using a Tabu Search algorithm. Ding et al. (2005) add the objective of minimizing ungated flights and directly solve the quadratic model using a hybrid Tabu Search and Simulated Annealing. Yan and Huo (2001) consider a different IP model minimizing walking distances and connection times. They propose a sensitivity analysis to reduce the number of variables.

Minimizing walking distance tends to concentrate traffic at the best located gates, which can lead to non robust solutions. Indeed, if a flight is delayed, its ground time is increased and it may overlap with the next operation assigned to the same gate. The robustness of an assignment plan is an important objective in the literature. Bolat (2000) proposes a model minimizing the

variance of idle times between two consecutive flights assigned to the same gate. He proposes a branch-and-bound algorithm and heuristic algorithms for solving the model. Lim and Wang (2005) propose a stochastic programming model that is transformed into a binary programming model to minimize the expected number of gate conflicts. They propose a hybrid meta-heuristic for solving their model. Yan and Tang (2007) propose a heuristic approach for minimizing flight delays due to gate blockages and reassignments. Their approach consists in iterating between two stages: a planning stage based on a multi-commodity flow network and a real-time stage based on simulations and reassignment rules for updating the planning stage. Diepen et al. (2012) suggest a column generation approach in order to establish robust assignment plan for Amsterdam Airport Schiphol (AAS). They identify gate types, i.e. groups of similar gates, and proceed in two phases. They first assign flights to the gate type through a column generation process aiming at generating good gate plans. Then a gate plan is assigned to each physical gate.

New objectives have appeared more recently in the literature. Dorndorf et al. (2008) take into account towing operations and shadow restrictions. They model the GAP as a Clique Partitioning Problem. Their model aims at simultaneously maximizing the total flight-gate affinity, minimizing the number of towing operations, minimizing the number of ungated flights and maximizing robustness by minimizing low buffers (idle time shorter than a given limit). A linear combination of these objectives is considered and the problem is solved by an ejection chain algorithm. Dorndorf et al. (2010) extend this model to minimize the deviation from a reference schedule. They suggest a method for building a reference schedule over a multi-period time horizon. Jaehn (2010) proposes a dynamic programming approach to solve a particular case of Dorndorf et al. (2008) where only flight-gate affinities are considered. He also proves the problem NP-hardness with a reduction from the optimal cost chromatic partition problem.

Kim et al. (2009) propose a new 0-1 QIP model for minimizing push back conflicts and taxi blocking. The model is further extended by Kim et al. (2013) to include the minimization of passenger transit times and baggage transport distances. They propose a tabu search and compare it to a linearization of the QIP and to a genetic algorithm for different airport configurations (parallel and horseshoe terminals) with randomly generated operations.

Genç et al. (2012) consider a new GAP, the objective of which is to maximize the total gate occupation time. Time is discretized in time slots of 5 or 10 minutes and the objective is to maximize the number of gate time slots used. They use a Big Bang Big Crunch method for solving instances from Istanbul Atatürk International Airport. Note that maximizing gate occupation time tends to reduce idle time at gates, which can lead to non-robust assignment plans.

In this paper, we consider the problem introduced by Dorndorf et al. (2008). The problem is referred to as the SAP since we consider both contact and remote stands. From a theoretical point of view, the SAP and GAP are equivalent. Our contributions to the SAP are summarized in what follows. We first prove that assigning each operation to a compatible stand is NP-complete based on a reduction from the circular arc graph coloring problem. As this corollary, it provides alternative proof for SAP NP-hardness compared with the proof given by Jaehn (2010). We also prove the NP-hardness of particular cases left open by Jaehn (2010). While the literature considers heuristic algorithms, we propose a strong mixed integer programming (MIP) formulation that solves to optimality real-size instances in reasonable computation times. We also introduce two heuristic algorithms based on spatial and time decompositions. In a numerical study, we compare our MIP-based approaches to the ejection chain algorithm described by Dorndorf et al. (2008) and to a simple greedy algorithm that mimics industrial practices (see Section 6). All methods are tested on real instances from two major European airports. MIP-based approaches are significantly better while computation times remain short enough for industrial purposes.

3. The stand allocation problem

In this section we formally introduce the Stand Allocation Problem (SAP) and the Stand Allocation Feasibility Problem (SAFP).

The ingredients for SAP can be summarized as follows:

- $O = \{1, .., m\}$ the set of operations. Operation $i \in O$ is defined by a start time a_i and an end time d_i, where $a_i < d_i$. a_i and d_i are assumed to be integers. Start and end times will often be referred to as the arrival and departure times.

- $S = \{1, .., n, n+1\}$ the set of stands. Stand $n+1$ is a dummy stand modeling unassignment, i.e. being assigned to stand $n+1$ is equivalent to being unassigned. We will also use the notation $\tilde{S} = S \setminus \{n+1\}$ for the set of real stands.

- $S_i \subset S$ the set of compatible stands for operation $i \in O$. Obviously, $n+1$ belongs to S_i for each operation $i \in O$. We will also use the notation $\tilde{S}_i = S_i \backslash \{n+1\}$.

- $U : O \rightarrow O \cup \{0\}$ the successor function. $U(i)$ is the direct successor of operation i for a given aircraft; i.e., if a turnaround is divided in two operations i and i', then $U(i) = i'$. Conventionally, if operation i does not have a successor then $U(i)$ is equal to 0. The end time d_i of an operation $i \in O$ is supposed to be equal to the start time $a_{U(i)}$ of its successor if there is one.

- $Q \subseteq O^2 \times S^2$ the set of shadow restrictions. If $(i, i', j, j') \in Q$ and operation i is assigned to stand j then operation i' cannot be assigned to stand j' and reciprocally.

- $c = (c_{ij})_{O \times S}$ the affinity matrix, i.e. c_{ij} is the affinity realized if operation $i \in O$ is assigned to stand j.

Shadow restrictions represent adjacency conflicts (e.g. two large aircraft cannot be simultaneously assigned to adjacent stands due to space limitations). It should be noted that the dummy stand $n+1$ is not concerned by either overlapping or shadow restrictions.

An assignment can be seen as a mapping $\mathcal{A}$ from the set of operations O to the set of stands S. The evaluation $f(\mathcal{A})$ of an assignment $\mathcal{A}$ is defined as

$$f(\mathcal{A}) = \alpha f_1(\mathcal{A}) - \beta f_2(\mathcal{A}) \tag{3.1}$$

where α and β are non negative and $f_1(\mathcal{A})$ and $f_2(\mathcal{A})$ are the total operation-stand affinity and the number of towing operations, respectively. Without loss of generality, we set $\alpha = 1$ in what follows.

The objective is to find an assignment maximizing $f(\mathcal{A})$ while respecting operation-stand compatibilities, shadow restrictions and overlapping constraints. In order to avoid assignment to the dummy stand, the affinity of an operation i for the dummy stand $n+1$ can be set to a high negative value.

Finally the Stand Allocation Feasibility Problem (SAFP) is the problem of determining whether there is a feasible solution not using the dummy stand.

4. Complexity of the stand allocation problem

In this section, we focus on a special case without successor relations ($U(i) = 0, \forall i \in O$ and $\beta = 0$) and without shadow restrictions ($Q = \emptyset$). An instance of the SAP can thus be denoted as $I(O, S, S_i, c)$. An instance of the associated feasibility problem SAFP is denoted as $I(O, \tilde{S}, \tilde{S}_i)$.

We first present the current complexity status of the SAP and highlight a number of open special cases. Then, we show how to formulate the SAFP as a graph coloring problem and prove its NP-completeness by a polynomial reduction from the circular arc graph coloring problem. Finally, we show the NP-hardness of a number of special SAP cases by polynomial reductions from SAFP.

4.1. Current complexity status and contributions

Jaehn (2010) proves that the SAP is NP-hard. His proof is based on a special case without compatibility constraints and where operations have the same affinity for each stand ($c_{ij} = c_j \in \mathbb{N}$). This case is proven NP-hard by a polynomial reduction from the Optimal Cost Chromatic Partition Problem (OCCP) in interval graphs. Kroon et al. (1997) prove that the OCCP in interval graphs is polynomial when c_j take at most 2 different values (e.g. $c_j \in \{0, 1\}$). They also show that it is NP-hard when c_j take at least 4 different values (e.g. $c_j \in \mathbb{N}$) while the problem is left open when c_j take exactly 3 different values (e.g. $c_j \in \{0, 1, 2\}$).

In the computational experiments (see Section 7), we consider three affinity functions that model different realistic situations : $c_{ij} \in \{0, 1\}$, $c_{ij} \in \{0, 1, 2\}$ and $c_{ij} \in \mathbb{N}$. Jaehn's proof does not provide a conclusion with respect to the complexity status when $c_{ij} \in \{0, 1\}$ or $c_{ij} \in \{0, 1, 2\}$. We will show that these special cases are also NP-hard. We will also prove that the SAP with compatibility constraints is NP-hard, for any of the above affinity functions.

Table 1 summarizes the results from the literature and our own contributions.

4.2. The stand allocation feasibility problem as a graph coloring problem

In this section, we show that the SAFP can be modeled by a graph coloring problem. Let $I(O, \tilde{S}, \tilde{S}_i)$ be an instance of SAFP. Let $G_I = (V \cup W, E)$ be an undirected graph where $V = \{v_1, .., v_n\}$ and $W = \{w_1, .., w_m\}$. Vertex v_j corresponds to stand $j \in \tilde{S}$ and vertex w_i corresponds to operation $i \in O$. To simplify matters, we will speak of stands and operations for vertices of V and W. The edges of the graph are defined as follows:

- $v_j v_{j'} \in E \quad \forall j, j' \in \tilde{S}$ such that $j \neq j'$,

- $w_i w_{i'} \in E \quad \forall i, i' \in O$ such that $i \neq i'$ and $[a_i, d_i[\cap [a_{i'}, d_{i'}[\neq \emptyset$, i.e. if operations i and i' overlap,

Affinity	Without compatibility constrains	With compatibility constraints
$c_{ij} \in \{0,1\}$	NP-hard (*)	NP-hard (*)
$c_{ij} \in \{0,1,2\}$	NP-hard (*)	NP-hard (*)
$c_{ij} \in \mathbb{N}$	NP-hard (Jaehn, 2010)	NP-hard (Jaehn, 2010)
$c_{ij} = c_j \in \{0,1\}$	P (Jaehn, 2010; Kroon et al., 1997)	NP-hard (*)
$c_{ij} = c_j \in \{0,1,2\}$	Open (Kroon et al., 1997)	NP-hard (*)
$c_{ij} = c_j \in \mathbb{N}$	NP-hard (Jaehn, 2010)	NP-hard (Jaehn, 2010)

Table 1: Complexity status of the stand allocation problem. (*) indicates the new results established in this paper.

- $v_j w_i \in E \quad \forall i \in O, j \in \tilde{S} \setminus \tilde{S}_i$, i.e. if operation i and stand j are incompatible.

Graphs G_I will be denoted as SAFP graphs. Figure 4 provides an example of such a graph.

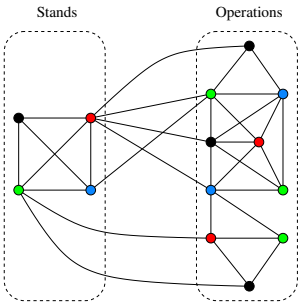


Figure 4: A 4-coloration of a SAFP graph

It should be noted that the graph induced by V is the clique K_n, thus G_I cannot be colored with less than n different colors. The graph induced by W is an interval graph. These subgraphs are linked by edges representing incompatibility constraints.

Property 1. *There is a feasible solution to an instance $I(O, \tilde{S}, \tilde{S}_i)$ of SAFP if and only if G_I admits a n-coloring.*

Proof. If G_I can be n-colored, then a feasible solution of I can be built from any n-coloring of G_I. Indeed, we assign each operation of a given color to the stand of the same color. Based on the construction of G_I, operations with the same color do not overlap and operations are compatible with the stand of the same color.

Conversely, if I is a feasible instance, then a n-coloring can be built from any feasible solution of I. A different color is assigned to each stand and each operation is colored with the color of the stand it is assigned to. Based on the construction of G_I, two adjacent nodes do not have the same color. □

This property implies that the SAFP and n-coloring problem of SAFP graphs have the same complexity status.

4.3. The circular arc graph coloring problem

The Circular Arc Graph Coloring Problem (CAGCP) was introduced and proven NP-complete by Garey et al. (1980). A brief overview of this problem is given below.

A circular arc A is a pair of positive integers (e, f) where e and f are different. Let $F = \{A_1, .., A_p\}$ be a set of circular arcs and k the maximum of all e_i and f_i ($k = \max\{e_i, f_i \mid A_i = (e_i, f_i), i \in \{1, .., p\}\}$). Consider a geometric arrangement of circular arcs as follows. A circle can be regarded as divided into k parts defined by k equally spaced points numbered clockwise as $1, 2, .., k$. In such a circle, each circular arc A_i previously defined can be regarded as representing an arc from point e_i to point f_i again in a clockwise direction. The span $Sp(A_i)$ of an arc $A_i = (e_i, f_i)$ is:

$$Sp(A_i) = \begin{cases} \{e_i + 1, .., f_i\} & \text{if } e_i < f_i \\ \{f_i + 1, .., k, 1, .., e_i\} & \text{if } e_i > f_i \end{cases}$$

Two arcs intersect if the intersection of their spans is not empty, i.e. $Sp(A_i) \cap Sp(A_j) \neq \emptyset$. Note that arcs do not intersect if they only share end points since the first point does not belong to the span.

We can define graph $G = (F, E)$, where $A_i A_j \in E$ if and only if A_i and A_j intersects. G is the circular arc graph induced by the set of circular arcs F. Figure 5 presents different representations of a circular arc graph.

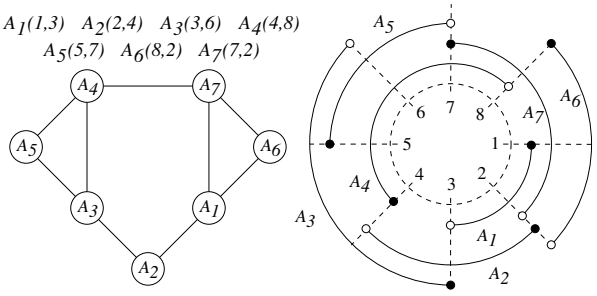


Figure 5: Three representations of a circular arc graph

CAGCP is the problem of finding a n-coloring for a circular arc graph. We will now show the relationship between this class of graph and our problem.

4.4. Complexity results

The NP-completeness of the SAFP can be shown by a reduction from the CAGCP.

Theorem 1. *The stand allocation feasibility problem (without shadow constraints and successor relations) is NP-complete.*

Proof. The SAFP is in NP as it represents a special case of a n-coloring problem. Let $F = \{A_i, .., A_p\}$ be a set of circular arcs and $G = (F, E)$ the circular arc graph induced by F. It is easy to show that the subgraph induced by $K = \{A_i \in F | e_i > f_i\}$ is a clique and the subgraph induced by $L = \{A_i \in F | e_i < f_i\}$ is an interval graph.

As K is a clique, G cannot be colored with less than $|K|$ colors. Hence, deciding whether G can be n-colored is polynomial if $n < |K|$. Coloring G is trivial if $n \geq p$. Hence, we assume that $n \in \{|K|, .., p-1\}$ in order to prove the above theorem.

We now build an instance $I(O, \tilde{S}, \tilde{S}_i)$ of the SAFP such that it accepts a solution if and only if G is n-colorable.

- For each circular arc $A_i = (e_i, f_i) \in L$, we define an operation i with the start time $a_i = e_i$ and end time $d_i = f_i$. As $A_i \in L$, $e_i < f_i$ and operation i is well defined.
- For each circular arc A_j of K, we define a stand. Stand j is compatible with operation i if and only if the associated arc A_j does not intersect the associated arc A_i.
- We add $n - |K|$ stands that are compatible with all operations.

Let G_I be the graph associated with I. It should be noted that G and G_I only differ by the vertices associated with the last $n-|K|$ stands. These vertices are only adjacent to other vertices of K. It follows that if G_I is n-colorable, so is G as G is a sub-graph of G_I. The reciprocal is valid because an n-coloring of G_I can be built from an n-coloring of G by assigning the $n - |K|$ colors not used in K to the $n - |K|$ last stands of G_I.

To conclude, G is n-colorable if and only if G_I is n-colorable. Hence coloring G_I is NP-complete. Together with Property 1, this implies the NP-completeness of the SAFP. □

As corollaries of Theorem 1, we now show that some special cases of the SAP, left open by Jaehn (2010), are NP-hard.

Corollary 1. *SAP with compatibility constraints and affinity coefficients verifying $c_{ij} = c_j \in \{0, 1\}, \forall i \in O, \forall j \in S$ is NP-hard.*

Proof. Since the SAFP is in NP and a solution can be evaluated in polynomial time, the SAP is in NP. Let us consider an instance $I(O, \tilde{S}, \tilde{S}_i)$ of the SAFP. We define the instance $I(O, S, S_i, c)$ of the SAP as follows:

- $S = \tilde{S} \cup \{|\tilde{S}| + 1\}$, i.e. $|\tilde{S}| + 1$ is the dummy stand,
- $S_i = \tilde{S}_i \cup \{|\tilde{S}| + 1\}$,
- $c_{ij} = \begin{cases} 1 & \forall i \in O, j \in \tilde{S} \\ 0 & \text{otherwise, i.e. for the dummy stand only} \end{cases}$.

$I(O, \tilde{S}, \tilde{S}_i)$ has a feasible solution if and only if $I(O, S, S_i, c)$ has a solution of value $|O|$. Furthermore, we define $I(O, S, S_i, c)$ such that $c_{ij} = c_j \in \{0, 1\}$. This proves the corollary. □

Corollary 2. *SAP without compatibility constraints and affinity coefficients $c_{ij} \in \{0, 1\}$ is NP-hard.*

Proof. As in Corollary 1, the SAP is in NP. Let us consider an instance $I(O, \tilde{S}, \tilde{S}_i)$ of the SAFP. We define the instance $I(O, S, S_i, c)$ of the SAP as follows:

- $S = \tilde{S} \cup \{|\tilde{S}| + 1\}$, i.e. $|\tilde{S}| + 1$ is the dummy stand,
- $S_i = S$ (no compatibility constraints),
- $c_{ij} = \begin{cases} 1 & \forall i \in O, j \in \tilde{S}_i \\ 0 & \text{otherwise} \end{cases}$.

$I(O, \tilde{S}, \tilde{S}_i)$ has a feasible solution if and only if $I(O, S, S_i, c)$ has a solution of value $|O|$. This proves the corollary. □

Corollaries 1 and 2 imply the new results presented in Table 1. They also provide alternative proof to the results of Jaehn (2010).

The NP-hardness of the special cases considered in this section does not mean that all instances are hard to solve. There may be constraints in industrial problems, making them easier to solve. Nevertheless, we did not identify such sub-structures in the instances considered in Section 7.

5. A mixed integer programming formulation

In this section, a first mixed integer program (MIP) formulation is presented. This model is then strengthened by reformulating a number of constraints and introducing new variables. Finally, an efficient process to break symmetries is presented.

5.1. A natural MIP formulation

Let us introduce the following decision variables:

- $x_{ij} = \begin{cases} 1 & \text{if operation } i \in O \text{ is assigned to stand } j \in S_i \\ 0 & \text{otherwise} \end{cases}$
- $y_i = \begin{cases} 1 & \text{if a towing operation is performed between operation } i \in O \text{ and its successor } U(i) \text{ if there is one} \\ 0 & \text{otherwise} \end{cases}$

Note that for the sake of simplicity, we define variables $x_{ij} = 0$ for each operation $i \in O$ and each non compatible stand $j \in S \setminus S_i$. Using these variables, the SAP can be formulated as follows:

$$\max \quad \sum_{i \in O} \sum_{j \in S_i} c_{ij} x_{ij} - \beta \sum_{i \in O} y_i \tag{5.1}$$

$$s.t. \quad \sum_{j \in S_i} x_{ij} = 1 \qquad \forall i \in O \tag{5.2}$$

$$x_{ij} + x_{i'j} \leqslant 1 \qquad \forall i, i' \in O, a_i \leqslant a_{i'} < d_i \quad \forall j \in S_i \cap S_{i'} \tag{5.3}$$

$$x_{ij} + x_{i'j'} \leqslant 1 \qquad \forall (i, i', j, j') \in Q \tag{5.4}$$

$$x_{ij} - x_{U(i)j} \leqslant y_i \qquad \forall i \in O, U(i) \neq 0, \quad \forall j \in S_i \tag{5.5}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in O, \forall j \in S_i \tag{5.6}$$

$$y_i \geqslant 0 \qquad \forall i \in O \tag{5.7}$$

MIP 1: A natural formulation for SAP

Constraints (5.2) ensure the assignment of each operation to one and only one stand. Constraints (5.3) prevent two overlapping operations from being assigned to the same stand. Constraints (5.4) guarantee that shadow restrictions are respected. Constraints (5.5) ensure that for each operation i towing is needed if the operation is assigned to stand j and not its successor $U(i)$. Note that, according to their definition, $y_i \in \{0, 1\}$ should be imposed. However, since $\beta \geqslant 0$ and since the objective function is maximized, we can simply impose $y_i \geqslant 0$ (5.7). Indeed, in any optimal solution, variable y_i will be set to the smallest value, i.e. 0 or 1 according to constraints (5.5) and (5.7).

5.2. A better MIP formulation

We will now strengthen this natural formulation by reformulating a number of constraints, introducing new variables and disrupting the objective function to break symmetries.

Strengthening overlapping and shadow constraints. Overlapping constraints (5.3) are weakly formulated and can be reformulated as follows. We introduce overlapping sets O_t as the set of operations overlapping time line t

$$O_t = \{i \in O \mid a_i \leqslant t < d_i\}$$

Overlapping constraints (5.3) can be replaced by

$$\sum_{i' \in O_{a_i}} x_{i'j} \leqslant 1 \qquad \forall i \in O, \forall j \in S_i \tag{5.8}$$

This formulation can be proven to be is ideal, i.e. describes the convex hull of integer solutions that satisfy overlapping constraints.

The same principle can be applied to strengthen shadow constraints. Constraint (5.9) is valid for any pair of stands $(j, j') \in S^2$ and any set of operations H and H' such that

- each pair of operations $(i, k) \in H$ overlap,
- each pair of operations $(i', k') \in H'$ overlap,
- there is a shadow restriction (i, i', j, j') between each operation $i \in H$ and each operation $i' \in H'$ on stands j and j'.

$$\sum_{i \in H} x_{ij} + \sum_{i' \in H'} x_{i'j'} \leqslant 1 \tag{5.9}$$

Nevertheless, the number of pairs of sets (H, H') suffers from a combinatorial explosion, even if only maximal sets are considered. We can heuristically aggregate the shadow constraints with the following algorithm.

While there are uncovered shadow restrictions $(i, i', j, j') \in Q$:

1. Let $H = \{i\}$ and $H' = \{i'\}$.
2. Complete set H: for each operation $k \in O$ (by increasing order of start time), add k to H if $(k, i', j, j') \in Q$ and if k overlaps each operation in H.
3. Complete set H': for each operation $k' \in O$ (by increasing order of start time), add k' to H' if for each operation $k \in H$, $(k, k', j, j') \in Q$ and k' overlaps every operation in H'.

$\mathcal{H}_{jj'}$ denotes the set of couples (H, H') generated by our algorithm for stands $j \in S$ and $j' \in S$.

Improving towing formulation. The linear relaxation can be strengthened by introducing variables

$$y_{ij} = \begin{cases} 1 & \text{if operation } i \in O \text{ is assigned to stand } j \in S_i \text{ and not its successor (if there is one),} \\ 0 & \text{otherwise.} \end{cases}$$

The objective function becomes

$$\sum_{i\in O}\sum_{j\in S_i} c_{ij}x_{ij} - \beta \sum_{i\in O}\sum_{j\in S_i} y_{ij}$$

and towing constraints (5.5) become

$$x_{ij} - x_{U(i)j} \leqslant y_{ij} \qquad \forall i \in O, \forall j \in S_i, U(i) \neq 0 \quad (5.10)$$

Indeed, it can be seen that the linear relaxation of both formulations has the same feasible domain in $x = (x_{ij})_{O\times S_i}$. Furthermore, for a given x, the optimal values of y variables in the linear relaxation are

- $y_i = \max_{j\in S_i}(x_{ij} - x_{U(i)j})$ for the first formulation,
- $y_{ij} = \max\{0, x_{ij} - x_{U(i)j}\}$ for the second formulation.

Consequently, $y_i = \max_{j\in S_i} y_{ij}$ and

$$\sum_{i\in O} y_i = \sum_{i\in O} \max_{j\in S_i} y_{ij} \leqslant \sum_{i\in O}\sum_{j\in S_i} y_{ij}$$

Therefore the formulation using variables y_{ij} is stronger.

Summary. To conclude, the problem can be reformulated as MIP 2.

$$\begin{aligned}
\max \quad & \sum_{i\in O}\sum_{j\in S_i} c_{ij}x_{ij} - \beta \sum_{i\in O}\sum_{j\in S_i} y_{ij} \\
s.t. \quad & \sum_{j\in S_i} x_{ij} = 1 && \forall i \in O \\
& \sum_{i'\in O_{a_i}} x_{i'j} \leqslant 1 && \forall i \in O, \forall j \in S_i \\
& \sum_{i\in H} x_{ij} + \sum_{i'\in H'} x_{i'j'} \leqslant 1 && \forall j, j' \in S, \\
& && \forall (H, H') \in \mathcal{H}_{jj'} \\
& x_{ij} - x_{U(i)j} \leqslant y_{ij} && \forall i \in O, U(i) \neq 0, \\
& && \forall j \in S_i \\
& x_{ij} \in \{0, 1\} && \forall i \in O, \forall j \in S_i \\
& y_{ij} \geqslant 0 && \forall i \in O, \forall j \in S_i
\end{aligned}$$

MIP 2: An improved formulation for the SAP

Breaking symmetries. If coefficients c_{ij} belong to a small set of values, this implies a high multiplicity of optimal solutions limiting the efficiency of branch-and-bound algorithms. For instance, some airports set c_{ij} to 1 for each contact stand and to 0 for each remote stand. A simple way to break symmetries is to disturb coefficients c_{ij}. We propose the following disruption that does not affect the optimal solution.

Property 2. *Assume that coefficients β and c_{ij} are integer (for all $i \in O$ and $j \in S_i$). Let γ_{ij} be arbitrary real numbers in $[0, 1)$ and $\delta_{ij} = \frac{\gamma_{ij}}{(m+1)}$.*

Thus, any optimal solution of the SAP with coefficients $c'_{ij} = c_{ij} + \delta_{ij}$ is also optimal for the SAP with coefficients c_{ij}.

Proof. Let f and f' be the objective functions of the original and disrupted problem. It should be noted that both problems have the same feasible solutions as they only differ by their objective functions. Let x be a feasible solution, then $f'(x) = f(x) + \varepsilon(x)$ with $\varepsilon(x) = \sum_{i\in O}\sum_{j\in S_i} \delta_{ij}x_{ij}$. We have $0 \leq \varepsilon(x) < 1$. Since coefficients β and c_{ij} are integers, $f(x)$ is also an integer and $\lfloor f'(x) \rfloor = f(x) + \lfloor \varepsilon(x) \rfloor = f(x)$.

Let x^* be an optimal solution for f'. For each feasible solution x we have $f(x) = \lfloor f'(x) \rfloor \leqslant \lfloor f'(x^*) \rfloor = f(x^*)$ and x^* is also optimal for f. □

In the numerical experiments, γ_{ij} is chosen randomly in [0,1) according to a uniform distribution.

6. Heuristic approaches

Regardless of how improved an MIP formulation can be, there exists instances that cannot be solved in a reasonable time. In this section, we present four heuristic algorithms that will be numerically compared to the exact MIP method in Section 7. The first two algorithms consist in splitting the problem into smaller sub-problems for which the MIP can be solved more quickly. The third algorithm is a greedy algorithm reflecting what was observed in practice in one of our partner airports. The fourth algorithm is the ejection chain algorithm designed by Dorndorf et al. (2008).

6.1. Spatial (or stand) decomposition

In the airports we work with, setting the affinity c_{ij} to 0 for remote stands is a reasonable assumption. This is not true for all airports since some remote stands might be preferable to others (e.g. short driving distance, stands that can be reached without a bus transfer, etc).

The stand decomposition method consists in splitting the set of stands into two disjunctive subsets. Subset B_1 contains stands with a positive affinity for at least one operation (typically contact stands). Subset B_2 contains the other stands with zero affinity for all operations (typically remote stands). Formally, we have $\tilde{S} = B_1 \cup B_2$ with $B_1 = \{j \in \tilde{S} : \exists i \in O, c_{ij} > 0\}$ and $B_2 = \{j \in \tilde{S} : \forall i \in O, c_{ij} = 0\}$.

We relax the assignment constraint (5.2) by

$$\sum_{j \in S_i} x_{ij} \leqslant 1 \qquad \forall i \in O$$

The relaxed problem provides an upper bound for the original problem. For the relaxed problem, not every operation may be assigned but any operation cannot be assigned more than once. The contribution of the stands in $B_2 \cup \{n+1\}$ is null or negative. As operations can be unassigned in the relaxed problem, this implies the following property:

Property 3. *The relaxed problem can be solved by considering the stands in B_1 only.*

This property reduces the size of the relaxed problem. We define the stand decomposition method in two phases:

- Phase I: solve the relaxed problem by considering stands in B_1 only,
- Phase II: fix the assignments defined in phase I and solve the SAP for the remaining operations and the stands in $B_2 \cup \{n+1\}$.

The upper bound provided in Phase I can be used to guarantee the solution a posteriori. The following property presents sufficient conditions under which the solution provided by the stand decomposition method is optimal for the original problem.

Property 4. *Conditions of optimality for the stand decomposition method.*
In Phase II, if each operation is assigned to a stand in B_2 without towing, the solution provided by the stand decomposition method is optimal for the original problem.

Proof. Under these conditions, the Phase II solution has the value 0 since the coefficient c_{ij} are all null for stands in B_2 and no towing operation is performed. Therefore, the global solution value is equal to the upper bound provided in Phase I. □

Property 4 can be used for solving Phase II in a more efficient way. Indeed, a Phase II solution with a 0 value is optimal (for Phase II). Consequently, if a heuristic algorithm provides such a solution, it is not necessary to solve a second MIP. In practice, we first apply the greedy algorithm presented in Section 6.3 and then solve the MIP only if the greedy algorithm fails to find a 0 value solution.

6.2. Time decomposition

The time decomposition consists in splitting the day into smaller intervals and iteratively solving the MIP for each sub-problem from the beginning of the day to the end of the day. Assignments decided in a previous iteration are not questioned in the current one except if the operation is still in progress. To reduce the total computation time, we split the day such that each sub-problem has almost the same size.

6.3. Greedy algorithm

The process of one of our partner airports is performed manually and is close to the following greedy algorithm.

1. Sort operations by increasing number of compatible stands.
2. Iteratively assign each operation to the compatible and available stand that maximizes the objective function. In case of multiplicity, choose the stands in lexicographic order.

The complexity of such an algorithm is in $O(m \log m + nm)$.

Once each operation has been assigned, the airport scheduler improves the solution by performing local changes. This process is similar to a descent algorithm using two types of moves : *simple* move (switch the assignment of an operation to another compatible and available stand) and *swap* move (swap the assignment of two operations). Only moves improving the objective are performed.

Such an algorithm ends very quickly in practice but it tends to fall into a local optimum that cannot be overcome as only improving moves are considered.

6.4. Ejection chain algorithm

An ejection chain algorithm is a local search metaheuristic where neighborhoods are defined not only by one move but by a sequence, or chain, of locally optimal moves. Performing more moves with each iteration is supposed to contribute to escaping the local optimum. Dorndorf et al. (2008) applied an ejection chain

algorithm to the stand allocation problem. We refer the reader to their paper for further details about their algorithm. In the next section, we compare our approaches to this algorithm, which has been replicated exactly.

7. Computational experiments

In this section, we compare the performance of the algorithms on realistic instances generated from the actual data of two major European airports. For the sake of privacy, these airports will be noted I and J.

7.1. Instances and tests environment

Computer. The results of mixed integer programs presented in this section were obtained using a Cplex 12.4 solver with default parameter tuning on a personal computer (Intel Core i5-2400 3.10Ghz, 4Go RAM) operating with Ubuntu 12.04 LTS operating system. Java Concert API was used to define the models.

Instances. Each instance corresponds to an operational day. For the largest airport, we have a single instance I. For the other airport, we have a test set $J = \{J_1, \cdots, J_{83}\}$ of 83 consecutive days. Table 2 presents characteristics of the instances with respect to the number of operations, the number of stands and the number of stands compatible with each operation.

Inst.	Ops	Simultaneous ops (peak)	Contact stands	Remote stands	Compatible stands (average)
I	703	92	43	122	131.2
Min J	397	37	60	49	34.8
Avg J	485	43	60	49	35.9
Max J	553	52	60	49	36.7

Table 2: Characteristics of the instances (Ops=Operations)

Operation-stand affinity. Pricing policies and performances measurements are complex substantially different from one airport to another. However, the operation-stand affinities c_{ij} can capture many practical situations. We will consider three affinity functions that represent different practices.

- *Passenger affinity:* Maximize the number of passengers assigned to contact stands

$$c_{ij} = \begin{cases} \text{number of passengers for operation } i \\ \text{if stand } j \text{ is a contact stand} \\ 0 \quad \text{otherwise} \end{cases}$$

- *Operation affinity:* Maximize the number of operations assigned to contact stands

$$c_{ij} = \begin{cases} 2 \quad \text{if operation } i \text{ is a whole turnaround} \\ \text{and stand } j \text{ is a contact stand} \\ 1 \quad \text{if operation } i \text{ is an arrival or a departure} \\ \text{operation and stand } j \text{ is a contact stand} \\ 0 \quad \text{otherwise} \end{cases}$$

- *Bus affinity:* Minimize the number of buses, which is equivalent to maximize the number of avoided buses

$$c_{ij} = \begin{cases} \text{Number of necessary buses for} \\ \text{operation } i \text{ if stand } j \text{ is a contact stand} \\ 0 \text{ otherwise} \end{cases}$$

The number of buses required is equal to the ceiling of the number of passengers involved in an operation divided by the capacity of a bus (80 in our numerical study). Note that we set affinity of a waiting operation at a contact stand to 0.

We use subscript $\_op$, $\_bus$ and $\_pax$ to indicate which affinity function is under consideration. For example, I_op corresponds to instance I with the operation affinity function.

Weighting of objectives. Coefficient $c_{i,n+1}$ is set to -10^6 to make the assignment of all operations the first priority. Note that all instances allow a feasible solution without using the dummy stand.

Coefficient β is respectively set to 1 for the optimization of operations at contact stands, 2 for optimization of buses and 100 for optimization of passengers. In this case both parts of the objective functions have similar weights.

Buffer time. We include buffer times of 10 minutes following the procedure presented in Section 1.

7.2. MIP 1 versus MIP 2

In this section, we evaluate the effect of strengthening constraints, towing reformulation and symmetry breaking, with respect to memory consumption, quality of the Linear Programming (LP) relaxation and computation times.

Table 3 shows that reformulating overlapping and shadow constraints substantially reduces the number of constraints. Note that the number of binary variables is the same since only continuous variables are added in MIP 2.

Instances		Gap ($z^*_{LP}/z^*_{MIP} - 1$) MIP 1	MIP 2 without y_{ij}	MIP 2	CPU time [s] MIP 1	MIP 2 without y_{ij}	MIP 2	MIP 2 + sym. break.
I_op		OOM	2.4%	0.0%	OOM	313.0	92.8	34.3
I_bus		OOM	3.5%	0.0%	OOM	1717.2	86.3	36.5
I_pax		OOM	2.8%	0.0%	OOM	512.2	74.3	28.2
J_op	Avg	3.2%	1.7%	0.0%	65.5	68.5	3.9	4.2
	Min	1.3%	0.8%	0.0%	2.7	1.6	1.0	1.3
	Max	5.8%	3.2%	0.1%	TL (0.0 %)	TL (0.0 %)	22.4	12.7
J_bus	Avg	4.8%	2.5%	0.0%	10.1	7.8	3.4	3.9
	Min	2.3%	1.2%	0.0%	3.1	1.7	1.0	1.6
	Max	6.7%	4.3%	0.1%	42.3	37.5	9.2	10.0
J_pax	Avg	4.1%	2.0%	0.0%	11.6	9.5	3.2	3.7
	Min	1.6%	0.9%	0.0%	3.0	1.6	1.0	1.2
	Max	5.8%	3.5%	0.1%	53.3	39.4	9.3	9.7

Table 4: Gap with the LP solution and computation time (OOM = Out of memory, TL (0.0 %) = Time limit of 1 hour reached, an optimal solution has been found but it cannot be proven because of remaining integrality gap)

Instance	Binary variables	Overlapping constraints MIP 1	MIP 2	Shadow constraints MIP 1	MIP 2
I	93 k	4.4 M	36 k	1.3 M	45 k
Avg *J*	17 k	315 k	10 k	146 k	6 k
Min *J*	14 k	196 k	8 k	93 k	4 k
Max *J*	20 k	415 k	11 k	195 k	7 k

Table 3: Number of binary variables and constraints

Table 4 presents the effect of the MIP formulation on the integrality gap and computation time, for the three affinity functions. A time limit of one hour is set.

We first discuss the results for the large instance (I) that cannot be solved with MIP 1 since the model definition phase exceeds the computer's memory. Reformulating overlapping and shadow constraints reduces memory consumption enough to be able to define the model. It also tightens the linear relaxation. Reformulating towing (i.e. replacing y_i by y_{ij}) further strengthens the linear relaxation and yields a zero integrality gap for most instances. MIP 2 without towing reformulation provides the optimal solution for all objectives within 5 to 30 minutes. Towing reformulation reduces the computation time to 1 minute and 30 seconds. The symmetry breaking method further reduces the computation time to approximately 30 seconds.

We now discuss the results for the medium-sized airport (J). MIP 1 does not exceed the available memory since the 83 instances are much smaller than I. The results with respect to the quality of the LP relaxation are similar to those of I. Furthermore, reformulating the constraints significantly improves the integrality gap, but there is little impact on computation times (probably because Cplex also uses an aggregation method based on cliques). While towing reformulation reduces computation times in a systematic and significant way, symmetry breaking has no effect on them.

These first numerical experiments show that the different reformulations strengthen the model and offer reasonable computational times for all instances and affinity functions under consideration. In what follows, only MIP 2 with symmetry breaking will be considered and will be simply referred to as exact MIP.

7.3. Comparison of algorithms

In this section, we compare the exact MIP method with the MIP decomposition methods (time and stand), the ejection chain algorithm and the greedy algorithm. For the time decomposition method, we split the day into three intervals for the large airport (I) and into two intervals for the medium-sized airport (J). We have tested other splits and found that these choices offer a good trade-off in terms of solution quality and computation times.

Table 5 presents the gap to optimality and the computation time for the three objective functions. The minimum, maximum and average values are presented for instance set J (83 instances).

On the one hand, Table 5 reveals that MIP based approaches provide significantly better solutions than the ejection chain and the greedy algorithms. The exact MIP always finds an optimal solution (and proves its optimality) in less than 40 seconds. The stand decomposition heuristic provides an optimal solution most of the time for both airports and the maximum gap is 0.3%.

Instances		Gap $(1 - z/z^*)$ MIP	SD	TD	EC	Greedy	CPU time [s] MIP	SD	TD	EC	Greedy
I_op		0.0%	0.0%	0.7%	5.0%	18.1%	34.3	20.6	16.3	1.7	0.1
I_bus		0.0%	0.0%	0.4%	6.9%	26.4%	36.5	37.8	20.6	4.1	0.1
I_pax		0.0%	0.0%	0.3%	6.8%	27.3%	28.2	21.1	12.1	3.0	0.1
	Avg	0.0%	0.0%	0.1%	2.0%	6.4%	4.1	2.6	2.7	0.4	<0.1
J_op	Min	0.0%	0.0%	0.0%	0.7%	3.4%	1.2	0.6	1.1	0.2	<0.1
	Max	0.0%	0.3%	0.4%	3.6%	19.9%	13.7	15.1	6.0	0.9	0.2
	Avg	0.0%	0.0%	0.2%	4.0%	7.9%	3.8	2.6	2.6	0.4	<0.1
J_bus	Min	0.0%	0.0%	0.0%	1.3%	3.8%	1.2	0.6	1.2	0.1	<0.1
	Max	0.0%	0.0%	0.7%	7.6%	13.2%	8.8	10.7	5.8	0.8	0.2
	Avg	0.0%	0.0%	0.1%	3.2%	6.5%	3.7	2.6	2.6	0.4	< 0.1
J_pax	Min	0.0%	0.0%	0.0%	1.3%	3.1%	1.2	0.6	1.2	0.1	<0.1
	Max	0.0%	0.0%	0.5%	5.9%	10.9%	9.8	9.4	4.5	0.8	0.2

Table 5: Comparison of the different methods (MIP=MIP2+ symmetry breaking, SD=Stand Decomposition, TD = Time Decomposition, EC= Ejection Chain)

The time decomposition heuristic provides very good solutions with gaps of less than 0.7%. The greedy algorithm offers poor performance for all instances and affinity functions, with a gap of up to 27.3 % for the large airport (I). The ejection chain algorithm outperforms the greedy algorithm with a gap of up to 7.6 % and an average gap of 2.0 % to 4.0 % for the medium-sized airport (J).

On the other hand, Table 5 shows that the greedy algorithm and the ejection chain are faster than the MIP based approaches. Nevertheless, the MIP based approaches offer reasonable computation times for industrial applications. They solve all instances of the medium-sized airport (J) in less than 15 seconds and in less than 40 seconds for the large airport (I). Regarding instances I and J, the stand decomposition method generally outperform exact MIP with respect to computation time, but its effect is sometimes more mixed. Time decomposition is the fastest MIP method with computation times approximately halved with respect to the exact MIP method. The differences between the exact MIP and the decomposition methods will be more significant when considering instances with more operations (see Section 7.4).

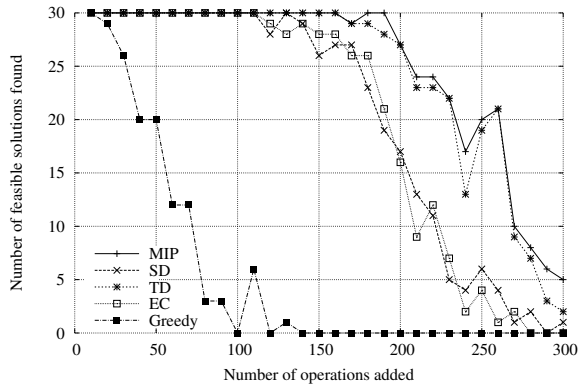
Our experiments lead us to conclude that MIP based approaches are suitable for solving the stand allocation problem for the set of instances considered. Indeed, they offer optimal or near-optimal solutions while ensuring reasonable computation times. The time decomposition method in particular offers the best trade-off between solution quality and computation time.

7.4. Feasibility

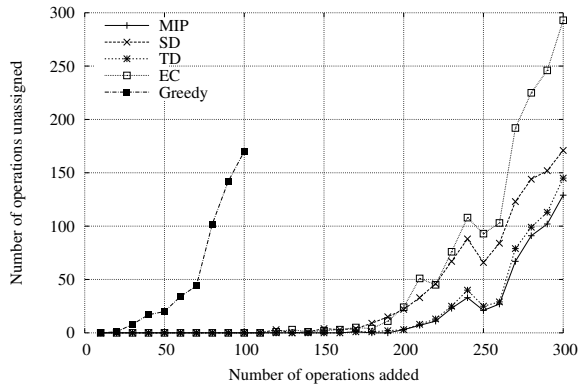
In Section 4, we show that deciding whether there is a feasible solution without the dummy stand is NP-complete. In this next section, we illustrate how important this result is from a practical point of view and compare the ability of each method to find a feasible solution when there is one. Obviously, any algorithm finding a feasible solution leads to the conclusion that an instance is feasible. However only exact methods, such as our MIP formulation, are able to guarantee that there is no feasible solution.

All the instances considered so far admit feasible solutions. In order to test the ability of each algorithm to find a feasible solution, we add a given number s of operations chosen randomly from the 82 other instances in J to the largest instance of J (553 operations). When an operation is added, we also add all the operations involved in the same turnaround while compatibility and objective coefficients are not changed. For each $s = 10, 20, \cdots, 300$, we simulate 30 instances. We then run the 5 algorithms for each of the 900 instances with the passenger affinity function. Note that the optimization is allowed to run to the end, i.e. it is not stopped when a feasible solution is found. Figure 6 presents the number of instances for which a feasible solution is found, the total number of unassigned operations and the CPU time for each algorithm.

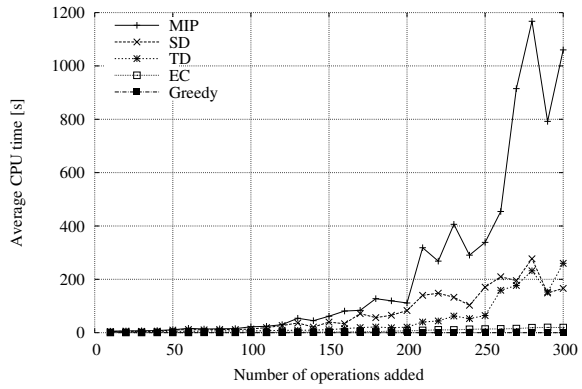
In Figure 6(a), we observe that the greedy algorithm begins to fail to find feasible solutions with only 20 additional operations. The ejection chain algorithm and the stand decomposition method handle all instances with up to 110 operations added while the exact MIP and the time decomposition method can go up to 160 operations. Figure 6(b) shows that the number of unas-



(a) Number of instances solved



(b) Number of instances solved



(c) Computation time

Figure 6: Effect of the number of operations on finding a feasible solution

signed operations with the time decomposition method is very close to the exact MIP method. On the contrary, the ejection chains fails for approximately twice as many operations. The number of unassigned operations grows very quickly for the greedy algorithm, which is why it has not been plotted.

In Figure 6(c), we observe that the MIP computation times grow exponentially with the number of operations but remain reasonable up to the addition of 250 operations. The time and stand decomposition methods suffer less from this phenomenon since the MIPs solved are smaller.

To conclude, the exact MIP or time decomposition methods are preferable for handling the most congested instances. Once again, the time decomposition algorithm offers the best trade-off in terms of computation time.

7.5. Passengers at contact stand versus number of towing operations

Maximizing operation-stand affinity contradicts the idea of minimizing the number of towing operations. A trade-off can be found by tuning coefficient β. However, choosing the values may prove to be a challenging task. This is why we propose to use Pareto curves to support the decision maker. Pareto curves translate the choice of abstract coefficients in terms of business measures. In this section, we consider the passenger affinity function since it provides smoother curves.

Figure 7 plots the Pareto curve linking the number of towing operations to the percentage of passengers assigned to contact stands. This curve was obtained by solving Instance I for 100 values of β, from 0.1 to 10 with a step of 0.1. Each point is Pareto optimal, i.e. not dominated by any other solution. A solution is said to dominate another one if it is better, or at least equal, for all objectives simultaneously.

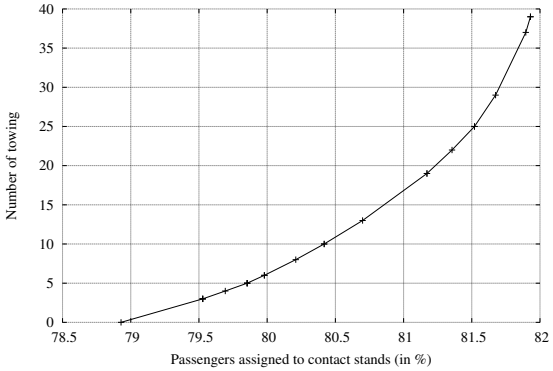


Figure 7: Passengers at contact stand versus number of towing operations for I

Plotting such a curve might be time consuming as the problem has to be solved several times. However, air traffic is mainly repetitive in the sense that it does not

significantly change from one week to the next. Therefore the coefficient values do not need to be discussed everyday and can be previously set with the help of Pareto curves for reference operational days.

The extreme left-hand point indicates the minimal number of towing operations. As it is Pareto optimal, it also provides the best possible percentage of passengers assigned to contact stands with this given number of towing operations. On the other hand, the extreme right-hand point indicates the maximal percentage of passengers that can be assigned to contact stands and the associated minimum number of towing operations.

We observe that the first three towing operations enable a 0.6% gain in passengers at the contact stand whereas fourteen additional towing operations are needed to gain the last 0.4% of passengers at the contact stand.

8. Conclusion and future prospects

In this paper, we prove that finding a feasible solution for the stand allocation problem is NP-complete. As a corollary, this proves the NP-hardness of the optimization problem. We then propose a strong MIP formulation and two heuristic algorithms. Our heuristic algorithms are based on the decomposition of the problem (spatial and temporal) where the sub-problems are solved using the MIP formulation. Based on instances from two European airports, we compare our approaches with the ejection chain method proposed by Dorndorf et al. (2008) and a greedy algorithm representing the current practice of a partner airport.

Computational experiments show that our MIP based approaches provide significantly better solutions than the other methods tested but need more computation time. Nevertheless, the computation times are short enough for an industrial application.

The instances considered in this paper come from large European airports. However, this does not necessarily mean that the proposed methods can be applied to the biggest airports in the world. Considering the reported computation times, the methods should be able to deal with bigger instances, and when this is not the case, our decomposition can be hybridized to handle the biggest instances: time decomposition can be applied to both phases of the stand decomposition heuristic. The stand decomposition can also be generalized in a terminal decomposition, as many flights are already preallocated to terminals in most airports.

Future research might focus on the aggregation of shadow constraints through clique constraints. The cliques used in this paper were generated heuristically but a theoretical study might lead to better ones being found and consequently stronger constraints.

Future research might also take robustness into more detailed consideration. Buffer times might be managed as an objective and not as a constraint, as Dorndorf et al. (2008) propose. Stochastic optimization and simulation can also be considered as proposed by Lim and Wang (2005) and Yan and Tang (2007).

It would also be interesting to study alternative objectives such as minimizing risky connections or the total walking distance of passengers. Finally, another topic to be researched further is the integration of airports' decision making problems, in particular the integration of the stand allocation problem together with other key airport resources such as runways (i.e. a sequencing problem) and tarmac space (i.e. a routing problem) as Kim et al. (2013) propose.

Acknowledgment

We would like to thank the associate editor and the three anonymous referees for their constructive comments, which significantly improved the exposition of this paper. We also thank Grégory Morel for giving us the link between the coloring of the stand allocation feasibility graph and the circular arc coloring problem. Finally we thank colleagues from Amadeus who provided instances, development support, airport knowledge and feedback.

AppendixA. Significant tow times

In this appendix, we explain how to extend our MIP formulation to the case where tow times are significant and cannot be reasonably included in operation processing times. Let $\tau_{jj'}$ be the tow time from stand j to stand j'. For the sake of simplicity, we assume that the tow time does not depend on the aircraft type. This assumption can be easily relaxed.

Consider an operation i that has a predecessor $U^{-1}(i)$, which implies that i is the successor of $U^{-1}(i)$. When the tow time is neglected, the start time of operation i is equal to the end time of its predecessor operation : $a_i = d_{U^{-1}(i)}$. With positive tow times, the starting time of operation i is equal to $a_i^{jj'} = d_i + \tau_{jj'}$ if operation $U^{-1}(i)$ is assigned to stand j and operation i is assigned to stand j'.

Based on this, consider two operations i and i' such that $d_i \leqslant d_{i'}$. We distinguish the case where i' does not have a predecessor ($U^{-1}(i') = 0$) and the case where i' has a predecessor ($U^{-1}(i') \neq 0$).

Case 1 : $U^{-1}(i') = 0$

In this case, operation i' does not succeed any other operation. Operations i and i' overlap if and only if $a_{i'} < d_i$. Hence the overlapping constraint is the same as (5.3) in MIP 1 :

$$x_{ij} + x_{i'j} \leqslant 1 \quad \forall i, i' \in O, U^{-1}(i') = 0, \\ \forall j \in S_i \cap S_{i'}, \\ a_{i'} < d_i \leqslant d_{i'}$$

Case 2 : $U^{-1}(i') \neq 0$

In this case, operation i' succeeds operation $U^{-1}(i')$. Operations i and i' cannot be assigned to stand j if $U^{-1}(i')$ is assigned to stand j' and $a_{i'}^{j'j} < d_i$, which leads to following overlapping constraints :

$$x_{ij} + x_{i'j} + x_{U^{-1}(i')j'} \leqslant 2 \quad \forall i, i' \in O, U^{-1}(i') \neq 0, \\ \forall j \in S_i \cap S_{i'}, \forall j' \in S_{U^{-1}(i')}, \\ a_{i'}^{j'j} < d_i \leqslant d_{i'}$$

References

A. Bolat. Procedures for providing robust gate assignments for arriving aircrafts. *European Journal of Operational Research*, 120(1): 63 – 80, 2000. doi: 10.1016/S0377-2217(98)00375-0.

G. Diepen, J.M. Van Den Akker, J.A. Hoogeven, and J.W. Smeltink. Finding a robust assignment of flights to gates at amsterdam airport schipol. *Journal of Scheduling*, 15:703–715, 2012.

H. Ding, A. Lim, B. Rodrigues, and Y. Zhu. The over-constrained airport gate assignment problem. *Computers and Operations Research*, 32(7):1867 – 1880, 2005. doi: 10.1016/j.cor.2003.12.003.

U. Dorndorf, A. Drexl, Y. Nikulin, and E. Pesch. Flight gate scheduling: State-of-the-art and recent developments. *Omega*, 35(3):326 – 334, 2007. doi: 10.1016/j.omega.2005.07.001.

U. Dorndorf, F. Jaehn, and E. Pesch. Modelling robust flight-gate scheduling as a clique partitioning problem. *Transportation Science*, 42(3):292–301, 2008. doi: 10.1287/trsc.1070.0211.

U. Dorndorf, F. Jaehn, and E. Pesch. Flight gate scheduling with respect to a reference schedule. *Annals of Operations Research*, 194:177–187, 2010.

M. Garey, D. Johnson, G. Mille, and C. Papadimitriou. The complexity of coloring circular arcs and chords. *SIAM Journal on Algebraic Discrete Methods*, 1(2):216–227, 1980. doi: 10.1137/0601025.

H.M. Genç, O.K. Erol, I. Eksin, and M.F. Berber. A stochastic neighborhood search approach for airport gate assignment problem. *Expert Systems with Applications*, 39:316–327, 2012.

A. Haghani and M. Chen. Optimizing gate assignments at airport terminals. *Transportation Research Part A: Policy and Practice*, 32(6):437 – 454, 1998. doi: 10.1016/S0965-8564(98)00005-6.

F. Jaehn. Solving the flight gate assignment problem using dynamic programming. *Z Betriebswirtsch*, 80:1027–1039, 2010. doi: 10.1007//s11573-010-0396-9.

S.H. Kim, E. Feron, and J.P. Clarke. Assigning gates by resolving physical confllicts. In *AIAA Guidance, Navigation and Control Conference*, Chicago, 2009.

S.H. Kim, E. Feron, and J.P. Clarke. Gate assignment to minimize passenger transit time and aircraft taxi time. *Journal of Guidance, Control, and Dynamics*, 36:467–475, 2013.

L.G. Kroon, A. Sen, H. Deng, and A. Roy. The optimal cost chromatic partition problem for trees and interval graphs. In *Graph-Theoretic Concepts in Computer Science*, volume 1197 of *Lecture Notes in Computer Science*, pages 279–292. Springer Berlin Heidelberg, 1997.

A. Lim and F. Wang. Robust airport gate assignment. In *ICTAI '05: Proceedings of the 17th IEEE international conference on tools with artificial intelligence*, 2005.

R. S. Mangoubi and D. F. X. Mathaisel. Optimizing gate assignments at airport terminals. *Transportation Science*, 19(2):173 – 188, 1985.

J. Xu and G. Bailey. The airport gate assignment problem: Mathematical model and a tabu search algorithm. In *Proceedings of the 34th Annual Hawaii International Conference on System Science*, 2001. doi: 10.1109/HICSS.2001.926327.

S. Yan and C.M. Chang. A network model for gate assignment. *Journal of Advanced Transportation*, 32(2):176–189, 1998. doi: 10.1002/atr.5670320204.

S. Yan and C. Huo. Optimization of multiple objective gate assignments. *Transportation Research Part A: Policy and Practice*, 35 (5):413 – 432, 2001. doi: 10.1016/S0965-8564(99)00065-8.

S. Yan and C.H. Tang. A heuristic approach for airport gate assignments for stochastic flight delays. *European Journal of Operational Research*, 180:547–567, 2007.