Expert Systems With Applications 238 (2024) 121953

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Integrated task scheduling and personnel rostering of airports ground staff: A case study

Paola Cappanera *, Leonardo Di Gangi, Matteo Lapucci, Giulia Pellegrini, Marco Roma, Fabio Schoen, Alessio Sortino

Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Firenze, Via di S. Marta 3, 50139, Firenze, Italy

ARTICLE INFO

Keywords:
Staff rostering
Task scheduling
Integrated TSPR
Airport personnel scheduling
Integer linear programming

ABSTRACT

In dealing with personnel management in companies, two fundamental planning issues have to be handled: staff rostering and activities assignment. These two problems have historically been treated separately, in a sequential way; however, it is evident how strongly they are tied to each other and that solution quality inevitably drops if this connection is not taken into proper account. For this reason, also taking advantage of the massive recent advances in software and hardware technologies, the integrated task scheduling and personnel rostering problem (TSPR) has been formalized along with suitable algorithmic approaches to tackle both planning stages altogether. In this paper, we describe how the peculiar and complex case of airport ground staff was handled in a scenario defined by real-world data from a large airport in Italy. Specifically, we show that the problem can be cast into a mixed integer linear programming model. We then show that, to make the problem computationally tractable, the introduction within the model of a set of suitable valid inequalities is crucial. Indeed, as opposed to the base model from the literature, the novel, improved formulation allowed to effectively obtain near-optimal solutions in reasonable time even for the considered large-scale and real-world scenario.

1. Introduction

The staff scheduling problem, also known in the literature as *Personnel Rostering* problem, consists in planning suitable work shifts for employees in (usually large) companies, e.g., hospitals, call centers, transportation companies, or hotels, so that the demand of required goods or services can be satisfied (Brucker et al., 2011; Cappanera et al., 2022; Özder et al., 2020; Van den Bergh et al., 2013). The optimal planning of work shifts may be highly beneficial both to employers, who can minimize costs, and to employees, whose preferences concerning daily timetables or days off may be satisfied.

The basic version of the staff scheduling problem requires to cover pre-defined shifts employing workers from a roster. The working hours of employees are typically bound by their contracts, which induce the main constraints in the optimization models. To complicate matters further, the staff scheduling problem may span a multi-day planning horizon and employees may be characterized by skills required to cover specific shifts (De Bruecker et al., 2015).

Recently, with the specialization and diversification of occupations in the work environment, staff scheduling problems where a worker has to perform several activities in the same shift are increasingly common, leading to even more complex problems. A second issue workforce planners have to deal with is thus that of *Task Scheduling*, i.e., the assignment of each activity to be carried out by a sufficient number of active, skilled enough workers to cover the demand (Kolen et al., 2007).

Historically, the two aforementioned problems have been addressed separately, in a sequential fashion, from the operations research community: first, the working shifts are constructed for each employee and then the tasks are assigned within the shifts. Clearly, not assigning tasks and shifts simultaneously eventually leads to lower-quality solutions (Ernst et al., 2004).

Of course, this kind of approach has been motivated by computational sustainability reasons. However, exploiting the increasing computational power provided by modern hardware and the algorithmic, software advances, the *integrated task scheduling and personnel rostering* (TSPR) problem was finally formalized and rigorously defined as a mixed-integer linear programming model by Smet et al. (2016).

Since then, more complex versions of TSPR have also started to be analyzed to meet specific application needs; for example, continuous training of employees has been considered in the context of inclusion

* Corresponding author.
E-mail addresses: paola.cappanera@unifi.it (P. Cappanera), leonardo.digangi@unifi.it (L. Di Gangi), matteo.lapucci@unifi.it (M. Lapucci), giulia.pellegrini1@stud.unifi.it (G. Pellegrini), marco.roma@unifi.it (M. Roma), fabio.schoen@unifi.it (F. Schoen), alessio.sortino@unifi.it (A. Sortino).

https://doi.org/10.1016/j.eswa.2023.121953
Received 12 July 2023; Received in revised form 29 September 2023; Accepted 1 October 2023
Available online 10 October 2023

of people with disabilities (Maheut et al., 2023); ordering and grouping constraints to satisfy customers' preferences were taken into account when dealing with employees of a wellness center (Mansini et al., 2023); grouping was also something to be handled in the case of residential care under pandemic conditions to mitigate the spread of infections (Moosavi et al., 2022).

We can thus observe that staff scheduling is a very context-specific problem: each case study has its peculiarities and complexities. This is of course also and particularly true for the case of ground staff in airports, which is the focus of the present work.

The examined instance, coming from a large airport in Italy, is indeed particularly hard to handle for several reasons. First, airports are a working environment operating 24 h a day, 7 days a week, where tasks repeat continuously and may even span two consecutive days. Second, employees (i) own skills making them almost unique in the team of workers (heterogeneous workforce), (ii) have legality and collective agreements to be satisfied, and (iii) typically, there is a mix of full and part-time contracts. Third, the planning horizon is made up of multiple days (typically a week) with a (highly) time-varying demand. Fourth, each employee can perform several different activities – of widely varying lengths, some of them even a few minutes long – during the same day-shift, and high flexibility is required in schedule definition.

The main contribution of the present paper is twofold: on the one hand, we show that Mixed-Integer Programming can be used with large-scale real-world instances of the TSPR problem; on the other hand, we present novel mathematical elements within the model that allow to make computation sustainable. Findings can be of general interest for other personnel scheduling problems involving task assignment.

The rest of the manuscript is organized as follows: in Section 2, the specifics of our case study are outlined; in Section 3, approaches from the literature for analogous problems are reviewed and discussed; in Section 4, our exact optimization model is defined in detail; in Section 5, we address a set of valid inequalities that allows to greatly improve the efficiency of the solution process; in Section 6, we report the results of computational experiments carried out on the real-world data at our disposal. Finally, in Section 7, we provide some concluding remarks.

2. Problem statement

The problem considered in this paper originates from the research project AIMS (Artificial Intelligence for the Management of Shifts) financed by the Tuscany Region. The project, whose participants came both from the academia and from industrial companies, aimed at the development of optimization tools for personnel shift scheduling based on activities. In particular, the partners of the project had the possibility of working with real data from a medium-sized airport in Italy; the objective was to build a set of optimization tools capable of taking into account all of the complexities of real-world scenarios. Thus, in the following, the definition of the problem, although being closely related to other similar problems found in the scientific literature, has been strongly influenced by the real case study on which the whole Research and Development project was based. The project was successfully concluded at the beginning of 2023 with a formal approval by the sponsoring partner.

Given a weekly planning horizon, a set of skilled employees, and a service demand to cover, the project described in this paper consists of devising an optimization model that defines shifts and assigns them to employees to satisfy demand. In the following, we will detail the key elements of the problem, i.e., service demand and employees.

In airports, which are open 24/7, some *activities* like "*check-in*" or "*vehicle cleaning*" must be done repeatedly depending on the arrival and departure times of flights. For each activity, and each day in the planning horizon, the demand for service is expressed in terms of the exact number of (skilled) employees required on a given time interval in a given *working area/section*. As an example, on Friday, for the check-in activity at the stopover employee area/gate B (working area/section) two employees, with proper skill, must be on duty from 11.25 to 11.45. As widely used in the literature, we use the term *task* to refer to these pieces of work that must be covered. So, there is a one-to-many relationship between activities and tasks: each activity corresponds to a set of tasks, and each task is associated with only one activity. Specifically, in our case study, service demand is described as a set of tasks to be covered, and each task is given by a tuple of information describing the time and location in which the corresponding activity must be performed, i.e., the day in the planning horizon, the time interval in the day – starting and ending time – and the working area; moreover, the set of skills required to perform that activity is specified. Tasks must be grouped to form *shifts*. A *shift* is defined as a time interval, characterized by a start time and a length. In the following, tasks are also referred to as *service requests* or simply *requests*.

Table 1
Request data.

Field	Description
`start`	Start date and time of request
`end`	End date and time of request
`qualification`	Necessary skill required to do the activity
`section`	When present, denotes the area where the request takes place
`demand`	Number of employees needed to satisfy the request

Table 2
Employee data.

Field	Description
`working_days`	Number of working days per week
`shift_duration`	Duration in hours of a working shift
`qualifications`	Skills of the employee
`section`	Specific area where the employee can operate

For each employee, data contains information on their skills (qualifications), the working area where they are qualified to work, and the contract. Contractual data defines the exact number of hours an employee must be available when on duty (*shift duration*), the maximum number of hours per week, and the maximum number of working days per week.

A detailed description of the available data is given in Tables 1 and 2.

Listed below, we summarize all the formal requirements specified for the airport considered in our study; note, however, that these specifics are common to the majority of main airports.

1. Every service request (task) must be satisfied with the exact number of appropriate employees.
2. Each employee can perform at most one shift per day.
3. The working hours per day of each employee must correspond to the contractual constraints.
4. The number of working days per week of each employee cannot be greater than a fixed number, defined in the contract.
5. For all employees, there must be a minimum number of hours between the end of a shift and the start of a shift on the next day.
6. Employees can perform only those activities they are qualified to do.
7. Two or more tasks cannot be performed by the same employee if they temporally overlap.

3. Literature review

The airline industry and its activities have drawn a major focus to operations researchers since the 1950s, giving rise to many challenging

optimization problems. Airports constitute one of the key pieces of this industry; their management involves the solution of many decision problems (Brusco et al., 1995; Cappanera & Gallo, 2004; Dowling et al., 1997; Nobert & Roy, 1998; Rong & Grunow, 2009), including personnel scheduling problems which is the focus of this work. A general description of the airport system was provided by Clausen and Pisinger (2011).

Literature on personnel scheduling problems is extremely vast; we refer the reader to the survey papers by Brucker et al. (2011), De Bruecker et al. (2015), Van den Bergh et al. (2013) and Özder et al. (2020) for thorough reviews of models and algorithms on this class of problems. In the context of airline companies, and more generally in public transport, a distinction is made between ground staff and traveling crew workers (airline pilots, train and bus drivers, stewards, etc.); the latter case has certainly received more attention in the literature (Deveci & Demirel, 2018; Herbers & Hromkovic, 2005; Kasirzadeh et al., 2017).

The problem addressed in this paper deals with the assignment of non-preemptive tasks and shifts. As outlined by Smet et al. (2016), few publications put the focus on this setting. Dowling et al. (1997), who incidentally focused on airport ground staff, addressed the problem as usual in two separate, sequential phases, to minimize over/understaffing, exploiting a simulated annealing strategy. Meisels and Schaerf (2003) addressed a timetabling problem where both tasks and possible shifts are known in advance; however, contrarily to general TSPR, only one task is assigned to each shift. Krishnamoorthy et al. (2012) proposed instead an algorithm to carry out multiple task assignment to multi-skilled employees, in the case where the shifts have been assigned in advance. Alternative approaches for this same problem have been proposed by Baatar et al. (2015) and Smet et al. (2014).

As outlined by Smet et al. (2016), the strong but yet complicated bond between task and shift assignments results in an integrated problem (TSPR) which has long been considered computationally impractical for realistically sized instances (see, e.g., Ernst et al., 2004). However, in the same paper, Smet et al. (2016) finally formalized the problem and showed it to be solvable, at least for synthetic problems, combining heuristic and decomposition strategies. Čalnerytė et al. (2020) also address the TSPR and propose a three-phase approach in which time is discretized in slots and a genetic algorithm and greedy heuristics are used in cascade to manage hard and soft constraints. More recently, Wang et al. (2023) proposed a clique-based formulation where, instead of assigning tasks directly to staff members as Smet et al. (2016) do, they assign tasks to shifts and shifts to staff members. In addition, they proposed two heuristics, respectively based on the rolling horizon concept and the iterative shift selection to solve medium to large real instances.

Problems with a structure similar to TSPR or having TSPR as a special case have also been studied recently. Campana et al. (2021) address a task and personnel scheduling problem arising in a large Italian company that provides cleaning services inside a hospital. For each task, the frequency with which it should be done inside the planning period is known as well as its duration and the skills required to perform it. A 3-phase approach is proposed which consists of the following steps: (i) for each task, determine the days on which it will be done; (ii) for each day, determine the sequence with which the tasks will be done; (iii) for each day and each operator, determine which sequence of tasks they will perform.

A general framework to address a plurality of staff scheduling problems has been recently proposed by Kletzander and Musliu (2020). A simulated annealing heuristic is implemented in the framework and the integrated TSPR problem defined by Smet et al. (2016) is used as a benchmark to solve instances with up to 40 employees. Compared with the results by Smet et al. (2016), the general approach seems to be able to find feasible solutions for more instances, while solution quality is not competitive in most cases.

Table 3
Notation — sets of data.

Set	Definition	Index
J	Set of employees	j
D	Set of days of the week	d
R	Set of requests	r
S	Set of shifts	s

Table 4
Notation - element-specific items.

Name	Definition
L_s	Duration of shift s
t_s	Starting time of shift s
S_j	Set of shifts assignable to employee j
H_j	Maximum hours per week for employee j
G_j	Maximum days per week for employee j
d_r	Number of employees required for request r
J_r	Set of employees qualified for request r

Summarizing, we can conclude that few studies have focused on the peculiar TSPR problem even after the formalization of the problem by Smet et al. (2016). Most of them are heuristics. When TSPR is dealt with using frameworks defined for more general problems, the performance obtained may not be satisfactory in terms of solution quality. This shortcoming therefore motivates the interest in studying approaches that exploit the particular structure of the problem. It is precisely into this literature gap that our work fits, proposing to enrich the clique-based mathematical models provided in the literature with crucial elements to make the problem computationally tractable even for large-scale real-world instances.

4. Problem formalization

In this section, we present the key elements of the approach we propose to address our case study.

4.1. Notation and basic concepts

As a first modeling element, we need to define a *shift*. A shift s is a time interval, characterized by a start time t_s and a duration L_s. We define a finite set S of possible shifts that employees can do. We discuss the choice of the possible starting times and shift durations, which strongly depend on the problem at hand, in Section 4.3. In the meantime, we assume that the set S is given.

In Tables 3 and 4 the notation for the considered problem is described, assuming the time horizon is one workweek.

In what follows, we describe the mathematical program, inspired by the TSPR model by Smet et al. (2016), used in this work to model our problem.

For ease of notation, we introduce two matrices: $A \in \mathbb{R}^{|R|\times|S|}$ and $B \in \mathbb{R}^{|R|\times|R|}$.

Matrix A is an incidence matrix, encoding the information about compatibility between request $r \in R$ and shift $s \in S$; each element $a_{r,s}$ is defined as follows

$$a_{r,s} = \begin{cases} 1 & \text{if request } r \text{ is time-wise contained in shift } s, \\ 0 & \text{otherwise.} \end{cases}$$

Matrix B is an incompatibility matrix, encoding the information about the incompatibility between two requests $r_i \in R$, $r_j \in R$; each element $b_{i,j}$ is defined as follows

$$b_{i,j} = \begin{cases} 1 & \text{if request } r_i \text{ is time-wise overlapped with request } r_j, \\ 0 & \text{otherwise.} \end{cases}$$

Matrix B can also be seen as an adjacency matrix of a graph where nodes represent requests and edges indicate incompatibility among

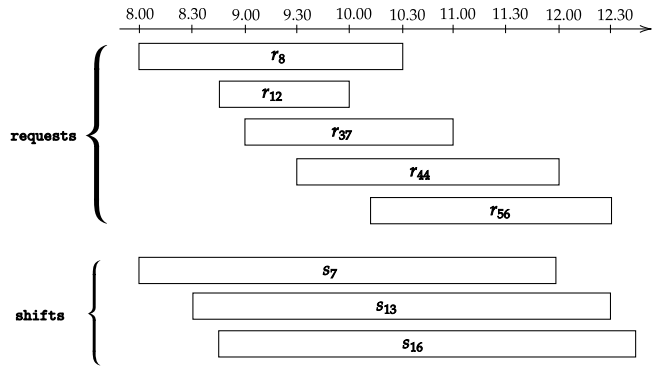

Fig. 1. Example of (part of) input data.

them. Cliques within this graph constitute sets of requests such that at most one of them can be assigned to a single employee. In the following, we will denote the set of maximal cliques in B as $C(B)$. An efficient algorithm to find all maximal cliques in B was proposed by Krishnamoorthy et al. (2012); we provide details about this procedure in Appendix.

The introduction of the above two matrices allows us to encapsulate the temporal element of the problem, reducing it to a static assignment task. Of course, the concept of incompatibility between two requests can be extended to further modeling requirements, such as spacial incompatibility (two requests cannot be performed by the same employee on the same day if their location is one far away from the other).

Example 4.1. To make it easier to understand the notation defined above and all the elements of the problem, we provide a toy example. In Fig. 1, a pictorial representation of a small set of requests to be satisfied and the set of possible shifts for a given day of the planning period are shown. Specifically, we are given 5 requests, i.e., $R = \{r_8, r_{12}, r_{37}, r_{44}, r_{56}\}$ and each request is represented as a horizontal rectangular box of length equal to its duration. For instance, request r_8 begins at 8.00 and has a duration of 2 h and a half. For that specific day, there are 3 possible shifts, i.e., $S = \{s_7, s_{13}, s_{16}\}$, and each shift is represented as a horizontal rectangular box of length equal to its duration. Shift s_7 begins, e.g., at 8.00 and has a duration of 4 h.

Fig. 2 shows respectively the compatibility matrix between requests and shifts (matrix A) and the incompatibility matrix between requests (matrix B). For example, request r_8 is time-wise contained in shift s_7, whereas it is not fully contained in shifts s_{13} and s_{16}, thus motivating the first row in matrix A (Fig. 2(a)). In addition, request r_8 is time-overlapped to all the other requests but request r_{56}, thus motivating the first row in matrix B (Fig. 2(b)).

We can now define the binary decision variables of the model:

$$X_j = \begin{cases} 1 & \text{if employee } j \text{ takes at least one shift,} \\ 0 & \text{otherwise,} \end{cases} \qquad \forall j \in J,$$

$$Y_{jsd} = \begin{cases} 1 & \text{if employee } j \text{ takes shift } s \text{ in day } d, \\ 0 & \text{otherwise,} \end{cases} \qquad \forall j \in J, \forall s \in S_j, \forall d \in D,$$

$$Z_{jr} = \begin{cases} 1 & \text{if employee } j \text{ performs request } r, \\ 0 & \text{otherwise.} \end{cases} \qquad \forall r \in R, \forall j \in J_r.$$

Before turning to the optimization model, we finally have to introduce for each shift s, the set I_s of shifts that are incompatible on the following day:

$$I_s = \{s_2 \in S \mid t_{s_2} + 24h - (t_s + L_s) \le \delta\},$$

where δ is an appropriate value, usually defined in the industry collective agreements. This set is needed to take into account that, usually, a minimum number δ of resting hours has to pass between two shifts taken consecutively by an employee.

4.2. The integrated TSPR model

We formulate our integrated TSPR as the following integer linear programming problem:

$$\arg\min_{X,Y,Z} \sum_{j\in J}\sum_{d\in D}\sum_{s\in S_j} L_s Y_{jsd} + \gamma \sum_{j\in J} X_j \tag{1a}$$

$$\text{s.t.} \sum_{j\in J_r} Z_{jr} \ge d_r \qquad \forall r \in R, \tag{1b}$$

$$\sum_{s\in S_j} Y_{jsd} \le 1 \qquad \forall j \in J, \forall d \in D, \tag{1c}$$

$$Z_{jr} \le \sum_{s\in S_j} a_{r,s} Y_{jsd} \qquad \forall j \in J, \forall r \in R, \forall d \in D, \tag{1d}$$

$$\sum_{r\in c : j\in J_r} Z_{jr} \le 1 \qquad \forall j \in J, \forall c \in C(B), \tag{1e}$$

$$\sum_{d\in D}\sum_{s\in S_j} Y_{jsd} \le G_j X_j \qquad \forall j \in J, \tag{1f}$$

$$\sum_{d\in D}\sum_{s\in S_j} L_s Y_{jsd} \le H_j X_j \qquad \forall j \in J, \tag{1g}$$

$$Y_{j,s_1,d} + Y_{j,s_2,d+1} \le 1 \qquad \forall j \in J, \forall s_1, s_2 \in S_j$$
$$\text{s.t. } s_2 \in I_{s_1}, \forall d \in D, \tag{1h}$$

$$Y_{jsd} \le X_j \qquad \forall j \in J, \forall s \in S_j, \forall d \in D, \tag{1i}$$

$$X_j \in \{0,1\} \qquad \forall j \in J, \tag{1j}$$

$$Y_{jsd} \in \{0,1\} \qquad \forall j \in J, \forall s \in S_j, \forall d \in D, \tag{1k}$$

$$Z_{jr} \in \{0,1\} \qquad \forall r \in R, \forall j \in J_r, \tag{1l}$$

where $\gamma > 0$ is a suitably defined constant.

Constraint (1b) forces requests covering: each request r has to be assigned to d_r workers. Constraint (1c) prevents employees from taking more than one shift per day. Moreover, by (1d) an employee is allowed to perform a request only if they are working at that moment. The assignment of incompatible requests to the same employee is avoided thanks to (1e). The contractual constraints of employees are enforced by (1f) and (1g). Constraint (1h) imposes the incompatibility condition between close shifts on consecutive days. By (1i), we model the fact that an employee can take a shift only if it is activated for the week. Finally, the (binary) domain of the variables is specified in (1j), (1k) and (1l).

As for the objective function, we minimize the idle time of the employees, i.e., the difference between the length of their shifts and the hours actually worked. To avoid unrealistic solutions where some employees work just one or two days throughout the week, we also add a penalty on the total number of active employees. The value of γ defines the trade-off between the two partly contrasting goals.

4.3. Identifying the shift set

The choice of the shift set S is crucial for the proposed solution. Indeed, a wide set of possible choices for the shifts may result in a too large number of binary variables to be handled with reasonable computational resources by software solvers. On the other hand, an excessively limited set of shifts may lead to the problem described in Section 4.3 having poor quality solutions and even being infeasible.

The possible length(s) of shifts usually depends on employees' contracts. Thus, the margin for choice mostly lies in the moments when shifts can be started. A reasonable option is to allow shifts to start every hour. Of course, with problems of moderate size, a denser set of possible shifts could be considered, to obtain solutions of better quality. On the other side, with particularly large problems a coarser grid might be necessary to reduce the computational effort up to sustainable levels.

s_7	s_{13}	s_{16}	A
1	0	0	r_8
1	1	1	r_{12}
1	1	1	r_{37}
1	1	1	r_{44}
0	1	1	r_{56}

(a) Request-shift compatibility matrix.

r_8	r_{12}	r_{37}	r_{44}	r_{56}	B
1	1	1	1	1	r_8
1	1	1	1	0	r_{12}
1	1	1	1	1	r_{37}
1	1	1	1	1	r_{44}
1	0	1	1	1	r_{56}

(b) Request-request incompatibility matrix.

Fig. 2. Matrices associated with data from Fig. 1.

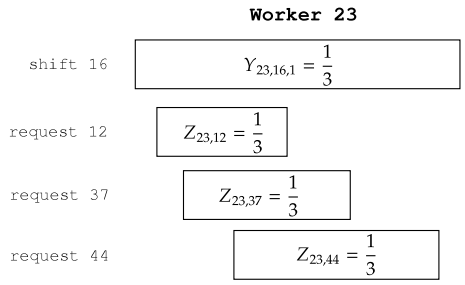

Fig. 3. Example of "unfeasible" fractional solution.

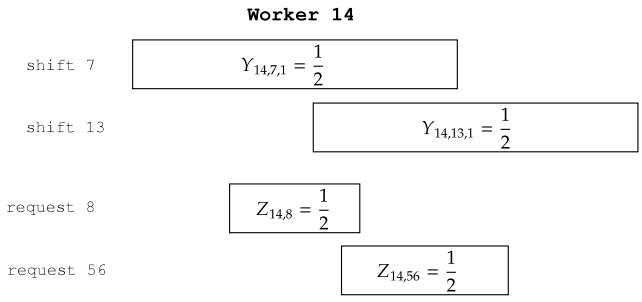

Fig. 4. Example of "unfeasible" fractional solution.

In any case, after the base selection of the shifts set, a further analysis has to be done based on the problem data. In particular, issues may arise with long-lasting tasks. Assume we let shifts start every hour (o'clock), and that the maximum duration of a shift is 8 h. Then, let r be a request of length 7 h 30 m starting at 9.40 and thus ending at 17.10. There is no shift of length 8 h starting at a time in $\{00.00, 01.00, \ldots, 23.00\}$ that fully contains the interval $[9.40, 17.10]$: the shift starting at 9.00 has to end at 17.00, before the request has been completed, whereas the following one starts at 10.00, when the request has already begun.

In general, starting times for shifts should be added so that all "long" requests can be covered.

5. Valid inequalities

To decrease computational solution times, valid inequalities can be introduced into the optimization model to improve the quality of the solution of the continuous relaxations.

Indeed, in the fractional solution situations occur that are impossible in feasible, binary, ones. The idea of valid inequalities is to try to add constraints that are satisfied by every feasible integer solution, and thus, redundant, but that are not satisfied at some vertices of the linear relaxation.

We begin this discussion by showing a situation that might be observed (and indeed it has been observed in our experiments) in the continuous relaxation of model (1). Consider the toy example shown in Fig. 1 and observe the situation depicted in Fig. 3.

In this example, worker $j = 23$ has been "partially" assigned to shift $s = 16$ in the fractional relaxation with a value of $\frac{1}{3}$. During this shift, this worker is also partially performing three tasks, $r = 12, 37, 44$, that are partly overlapping. It is evident that this is not possible and, indeed, in the optimal, binary-valued, solution this situation cannot happen. But in the relaxed problem, this is a feasible solution, as constraints (1d) and (1e) are satisfied.

In fact, constraint (1e) forbids assigning incompatible tasks to the same worker. But the constraint just asks that the sum of binary variables is no larger than 1 and this, as we can see, is satisfied in this case. What is lacking in the model is a constraint that links Y and Z variables for incompatible requests. In this example, we would like to add a constraint like

$$Z_{23,12} + Z_{23,37} + Z_{23,44} \leq \sum_{s\,:\,a_{12,s}+a_{37,s}+a_{44,s}\geq 1} Y_{23,s,1},$$

where the sum concerns all shifts that cover at least one of the requests 12, 37, 44. Generalizing:

$$\sum_{r\in c\,:\,j\in J_r} Z_{jr} \leq \sum_{s\in S(j,c)} Y_{jsd} \tag{2}$$

where c is a (maximal) clique of incompatible requests.

Given a clique c, the left-hand side sums, similarly as in constraint (1e), all of the assignments of those tasks to the same employee j. In the example, this sum is $\frac{1}{3} + \frac{1}{3} + \frac{1}{3} = 1$. The right-hand side considers, for the day in which the tasks have to be performed, all of the possible shift assignments of employee j to a shift which covers any of the tasks in the clique; the set of shifts covering at least one of the tasks in c is denoted by $S(j, c)$, i.e.,

$$S(j,c) = \left\{ s \in S_j \mid \sum_{r\in c} a_{r,s} \geq 1 \right\}.$$

In our case, this sum is $\frac{1}{3}$. Changing the constraint (1e) into (2), the above solution is immediately cut off.

Another case of "unfeasible" fractional solution can be observed in the example in Fig. 4, again referred to the toy example instance in Fig. 1.

Here, the assignment of worker $j = 14$ on day $d = 1$ has been split among two shifts, $s = 7$ and $s = 13$. Then, requests 8 and 56 can both be (partly) assigned to worker j without breaking constraint (1d) nor (1e). Note that the addition of constraint (2) does not cut this solution

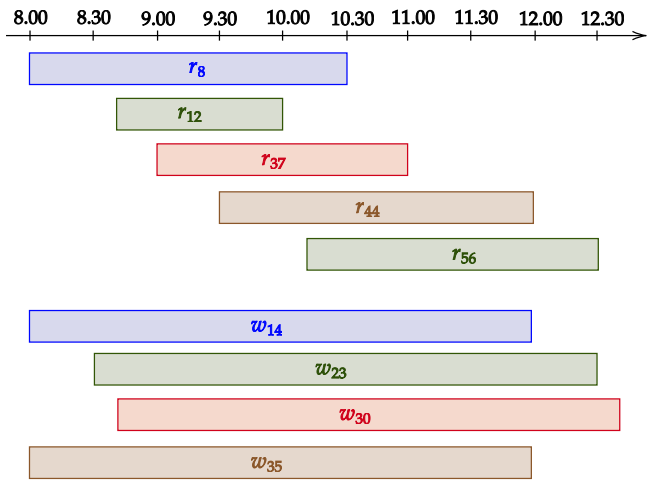

Fig. 5. An optimal solution for the toy problem from Fig. 1. The assignment of tasks to shifts and the staff rostering is color-coded. Worker w_{23} has been assigned to shift s_{13} and covers requests r_{12} and r_{56} (color-coded in green). The other three workers, namely w_{14} (blue), w_{30} (red), and w_{35} (brown) cover a request each and they have been assigned respectively to shifts s_7, s_{16}, and s_7.

off, as $1 = Z_{14,8} + Z_{14,56} = Y_{14,7,1} + Y_{14,13,1} = 1$. However, the relaxation bound can be strengthened. Since the overlapping requests 8 and 56 cannot be performed by the same worker, we notice that at least two fully active shifts (from different employees) need to be active to cover both. Hence, we should include in the model a constraint like

$$\sum_{j \in J_8 \cup J_{56}} \sum_{\substack{s \in S_j \\ a_{8,s}+a_{56,s} \geq 1}} Y_{j,s,1} \geq 2.$$

Generalizing, we can insert into model (1), for any (maximal) clique $c \in C(B)$ of incompatible requests, the constraint

$$\sum_{j \in \bigcup_{r \in c} J_r} \sum_{\substack{s \in S_j \\ \sum_{r \in C} a_{r,s} \geq 1}} Y_{jsd} \geq \sum_{r \in C} d_r, \tag{3}$$

where we are also taking into account the number of operators required to satisfy each request.

We conclude the section reporting, for the toy problem shown in Fig. 1, an optimal solution of the corresponding mathematical model. Fig. 5 shows that 4 workers are needed to cover the 5 requirements in R.

6. Experimental results

In this section, we describe the insights gathered dealing with our real-world case study, presenting and discussing the obtained results.

6.1. Case study

Our case study concerns a major international airport in Northern Italy for which we collected 10 instances, corresponding to different weeks spanning from 2022 to 2023.

The relevant information available for each request and each employee corresponds to that reported in Tables 1 and 2 respectively. Each employee is associated with a set of skills, and each request specifies one or more skills for any employee to be assigned to the corresponding task. Three instances include requests associated with a specific section within the department; in these cases, employees are also associated with (possibly multiple) sections; to handle this kind of specification, the combinations of skill and section can be considered as a whole specific skill. Overall, 58 possible skills can be associated with the employees.

We report some statistics for the 10 instances in our benchmark in Table 5. Specifically, for each instance we can find: the number of employees, the average number of skills for each employee, the number of requests, descriptive statistics (average, minimum, and maximum values) about the number of employees skilled for a request, same descriptive statistics about number of requests that employees own the skill to perform, and minimum and maximum number of daily requests.

For all employees in each test instance, the maximum number of working days per week is 5. Employees are associated, by their contracts, with shifts of length 4, 6, or 8 h; aggregate information about this data is reported in Table 6.

6.2. Experimental setup and evaluation metrics

Each problem instance is modeled and implemented according to formulation (1). The values of G_j and H_j for the jth employee, L_s and t_s for shift s, and d_r and the set J_r for request r are directly determined by instance data. The value of the trade-off parameter is set to $\gamma = 50$.

As for the set S_j of feasible shifts for employee j, we followed the methodology described in Section 4.3: first, we define shifts starting every hour o'clock. Then, it can occasionally happen that a request cannot be covered by any of the already available shifts; in those cases, a new shift option, starting exactly at the request start time, is generated.

For each instance, we solve 4 variants of model (1) (hereafter called configurations). Specifically, model (1) is run:

(i) without the addition of any valid inequality,
(ii) adding to the model only the set of inequalities (2),
(iii) adding to the model only the set of inequalities (3),
(iv) adding both sets of valid inequalities.

The solver is enabled to carry out its presolve phase. For each model variant, however, we also solve the pure continuous relaxation, to assess the effect of the presolve in terms of performance.

To provide a condensed view of the results, in the following we are making use of performance profiles (Dolan & Moré, 2002). Performance profiles provide a unified view of the relative performance of the solvers on a suite of test problems. Formally, consider a benchmark of $\mathcal{P}$ problem instances and a set of solvers $\mathcal{S}$. For each solver $\sigma \in \mathcal{S}$ and problem $\pi \in \mathcal{P}$, we define

$c_{\pi,\sigma}$ = the cost for solver σ to solve problem π,

where cost is the performance metric we are interested in. In particular, we will be interested in CPU time. We then consider the ratio

$$\eta_{\pi,\sigma} = \frac{c_{\pi,\sigma}}{\min_{\sigma \in S}\{c_{\pi,\sigma}\}},$$

which expresses a relative measure of the performance on problem π of solver σ against the performance of the best solver for this problem. If a solver fails to solve a problem, we shall put $\eta_{\pi,\sigma} = \eta_M$, with $\eta_M \geq \max\{\eta_{\pi,\sigma} \mid \pi \in \mathcal{P}, \sigma \in \mathcal{S}\}$.

Finally, the performance profile for a solver σ is given by the function

$$\rho_\sigma(\tau) = \frac{1}{|\mathcal{P}|} \cdot \left|\{\pi \in \mathcal{P} \mid \eta_{\pi,\sigma} \leq \tau\}\right|,$$

which represents the estimated probability for solver σ that the performance ratio $\eta_{\pi,\sigma}$ on an arbitrary instance π is at most $\tau \in \mathbb{R}$. The function $\rho_\sigma(\tau) : [1, +\infty] \to [0, 1]$ is, in fact, the cumulative distribution of the performance ratio.

Note that the value of $\rho_\sigma(1)$ is the fraction of problems where solver σ attained the best performance; on the other hand, $\lim_{\tau \to \eta_M^-} \rho_\sigma(\tau)$ denotes the fraction of problems solved from the given benchmark.

In addition to performance profiles, we will also make use of the cumulative distribution of absolute gaps for a given metric v; in

Table 5
Statistics for the 10 instances.

	#emp	#skills	#req	#emp/req			#req/emp			Daily #req	
		avg		avg	min	max	avg	min	max	min	max
1	170	11.4	2045	137.9	17	163	1658.4	7	2003	253	325
2	83	9.0	1292	75.2	51	83	1170.6	266	1292	179	190
3	87	8.8	1243	76.6	52	87	1094.2	212	1243	174	182
4	88	8.7	1248	77.5	52	88	1098.6	212	1248	149	193
5	88	8.7	1297	77.4	52	88	1141.2	212	1297	182	189
6	88	8.7	1274	77.5	52	88	1121.9	212	1274	175	190
7	88	8.7	1268	77.5	52	88	1116.7	212	1268	170	192
8	184	10.6	2116	129.4	18	177	1488.0	7	2074	266	367
9	108	6.9	1217	83.1	9	98	936.9	0	1217	154	197
10	169	11.3	2053	131.0	16	162	1591.6	21	1922	240	332

Table 6
Percentage of employees associated with a given shift length for each problem instance.

Shift length			
	4 h (%)	6 h (%)	8 h (%)
1	0	0	100
2	17	22	61
3	14	17	69
4	17	17	66
5	17	17	66
6	17	17	66
7	17	17	66
8	0	0	100
9	6	0	94
10	0	0	100

particular, this tool has a similar concept as performance profiles and is obtainable setting

$$\eta_{\pi,\sigma} = \frac{|v_{\pi,\sigma} - \text{opt}_{\sigma\in S}\{v_{\pi,\sigma}\}|}{\text{opt}_{\sigma\in S}\{v_{\pi,\sigma}\}},$$

where the opt operator denotes the minimum or the maximum according to the metric v selected. Distribution of relative gap is particularly useful when evaluating results in terms of objective values. Of course, in this case, we have $\rho_\sigma(\tau) : [0, +\infty] \to [0, 1]$.

As performance metrics of our experiments, we are interested in the following values:

- the optimal value of the problem continuous relaxation, denoted by f_{rel};
- the optimal value at root relaxation (when the presolve step is enabled), denoted by f_{root};
- the objective value of the first feasible solution encountered during the optimization process, denoted by f_f;
- the objective value of the best feasible solution encountered during the optimization process, denoted by f_{best};
- the runtime required to solve the relaxation, find a feasible solution, and find the best solution, denoted respectively by t_{rel}, t_f, t_{best}.

The notation is entirely summarized in Table 7.

6.3. Results

Figs. 6, 7 and 9 respectively show the cumulative distributions of the relative gaps for the metrics f_{best}, f_f, and f_{root} and f_{rel}; the two latter metrics are coupled in the same plot, sharing the reference optimal values, to appreciate the impact of the presolve step. Fig. 8 shows the performance profile for the t_f metric.

Table 8 provides an overview of the experimental results. For each instance, we compare the numerical results concerning the objective values f_{rel}, f_{root}, f_f, and f_{best}, the best lower bound value l_B and computation times t_{rel}, t_f and t_{best} related to the four configurations of

Table 7
Summary of symbols.

VI	Valid Inequalities
(P)	Presolve used
f_{rel}	Optimal objective value of relaxed model
f_{root}	Objective value of root relaxation
f_f	Objective value at first feasible solution encountered
f_{best}	Objective value at best solution encountered
t_{rel}	Time to solve continuous relaxation
t_f	Time to find a feasible solution
t_{best}	Time to reach best solution
l_B	Best lower bound value when solver stops
%gap	Optimality gap when solver stops

the model obtained by including or not the proposed valid inequalities. Times are expressed in CPU seconds. In addition, for each instance and each configuration, Table 8 reports the percentage relative gap (%gap) between f_{best} and l_B.

It can be noticed that the introduction of valid inequalities allows to obtain relatively small gaps. Instance 9 was solved to optimality (%gap = 0) by the model that includes VI (3); in this case, the running time was 30 026.05 s. In all the other cases, the model returned either a suboptimal solution with a (usually small) gap or no feasible solution within the time limit (a '–' in Table 8 indicates the absence of the value).

From the results it clearly emerges that the proposed valid inequalities are significantly effective. In fact, the model with none of them never returned a feasible solution within the time limit, while the addition of any of the proposed sets of inequalities allowed the model to return at least a feasible solution for all the instances except one case — instance 8 using VI (3).

Table 9 shows for each configuration the number of instances for which a specific configuration is ranked first (best), second, third, or fourth (worst) in terms of both objective value f_{best} and best-bound value l_B yielded. It can be noticed that, if we consider just the objective value, configuration (2) dominates the others, performing slightly better than (2)–(3). The cumulative distribution presented in Fig. 6 shows that configuration (3) is dominated by these other ones. Moreover, we can observe that each time configuration (3) found the best solution, it was always equaled by configuration (2). These considerations suggest that VI (2) is more effective than VI (3) to get the best objective value.

Instead, if we consider the best-bound value l_B reported in Table 9, configuration (3) dominates all the others; we can also observe the same behavior in Fig. 9, where the cumulative distribution corresponding to configuration (3) dominates that referred to configuration (2).

Considering the objective value of the first feasible solution f_f found by each configuration, the cumulative distributions plotted in Fig. 7 show that (2) dominates the others; also (2)–(3) has a better performance than (3), confirming that set of inequalities (2) leads the solver to reach a good quality solution even when solving the instance to the optimum is not possible.

Table 8
Results of the computational experiments on the 10 problem instances.

$\mathcal{P}$	VI	Objective value				l_B	%gap	Time (s)		
		f_{rel}	f_{root}	f_f	f_{best}			t_{rel}	t_f	f_{best}
1	No	1535.67	1 842.8	–	–	2 038	–	689.1	–	–
	(2)	13 122	13 122.0	13 680	13 680	13 122	4.1	996.4	80 542	80 542
	(3)	13 176	13 176.0	14 720	13 680	13 194	3.6	1 058.1	18 731	59 860
	(2)–(3)	13 218	13 218.0	15 210	13 738	13 248	3.6	2 062.8	61 660	66 942
2	No	910	1 387.0	–	–	3 620	–	61.7	–	–
	(2)	4791	4 794.0	6 976	4 902	4 814	1.8	106.2	5687	77 915
	(3)	4786	4 802.0	6 992	5 024	4 806	4.3	116.8	2208	99 669
	(2)–(3)	4791	4 794.0	6 990	4 894	4 814	1.6	210.1	6229	79 108
3	No	1554	1 558.4	–	–	2 310	–	83.5	–	–
	(2)	5028	5 028.0	5 862	5 106	5 032	1.4	229.3	8847	79 797
	(3)	5028	5 028.0	7 368	5 128	5 068	1.2	176.0	4504	63 182
	(2)–(3)	5028	5 028.0	7 330	5 106	5 032	1.4	344.2	9107	81 904
4	No	1554	1 567.1	–	–	2 520	–	84.6	–	–
	(2)	5018	5 018.0	6 016	5 128	5 022	2.1	194.8	14 183	70 914
	(3)	5018	5 018.2	7 424	5 146	5 108	0.7	179.9	6267	53 227
	(2)–(3)	5018	5 018.0	7 370	5 130	5 020	2.1	327.8	8726	66 433
5	No	1570	1 584.0	–	–	2 334	–	86.8	–	–
	(2)	5154	5 154.0	6 112	5 218	5 172	0.9	195.8	14 855	83 639
	(3)	5154	5 155.6	7 462	5 232	5 202	0.6	197.3	6512	96 570
	(2)–(3)	5154	5 154.0	5 914	5 218	5 170	0.9	333.8	8885	76 228
6	No	1554	1 564.3	–	–	2 230	–	81.7	–	–
	(2)	5061.25	5 061.3	5 808	5 226	5 096	2.5	200.1	24 856	100 000
	(3)	5060	5 060.0	7 324	5 310	5 138	3.2	214.2	4936	28 785
	(2)–(3)	5061.25	5 061.3	5 736	5 166	5 092	1.4	386.5	19 849	58 931
7	No	1554	1 566.3	–	–	2 270	–	83.5	–	–
	(2)	5110	5 110.0	5 826	5 246	5 162	1.6	207.8	27 259	96 643
	(3)	5110	5 110.0	7 458	5 246	5 214	0.6	212.8	8383	80 930
	(2)–(3)	5110	5 110.0	5 984	5 230	5 146	1.6	368.6	19 714	97 345
8	No	1524.53	2 220.7	–	–	3 338	–	826.5	–	–
	(2)	12 852	12 852.0	15 718	13 108	12 852	2.0	13 820.7	63 288	77 174
	(3)	12 852	12 852.0	–	–	12 852	–	1 509.4	–	–
	(2)–(3)	12 852	12 852.0	16 374	14 816	12 852	13.3	2 243.0	46 028	63 262
9	No	1610	2 058.0	–	–	3 060	–	65.3	–	–
	(2)	5748	5 748.0	8 422	5 788	5 748	0.7	231.1	2592	12 964
	(3)	5748	5 748.0	8 770	5 788	5 788	0	165.8	1973	11 024
	(2)–(3)	5748	5 748.0	6 344	5 788	5 748	0.7	354.2	519	1802
10	No	1719.18	2 362.5	–	–	4 188	–	560.1	–	–
	(2)	11 862	11 862.0	12 208	12 208	11 862	2.8	870.6	34 556	34 556
	(3)	11 862	11 862.0	12 674	12 208	11 898	2.5	732.1	17 937	25 429
	(2)–(3)	11 862	11 862.0	13 122	12 208	12 070	1.1	1 846.1	14 006	17 864

Table 9
Ranking of the models based on the objective value and the best bound: for each configuration of the model the number of instances for which it placed first, second, third, and fourth is reported.

VI	f_{best}				l_B			
	1st	2nd	3rd	4th	1st	2nd	3rd	4th
No	0	0	0	10	0	0	0	10
(2)	7	3	0	0	0	6	4	0
(3)	3	1	5	0	6	2	2	0
(2)–(3)	7	2	1	0	3	0	7	0

On the contrary, it can be observed in Fig. 8 that the configuration that uses the set of inequalities (3) yields feasible solutions of reasonable quality (the gap is relatively small anyway) faster than the other methods except one case where it does not find any. Moreover, we observe that adding this set of inequalities to the model together with (2) seems to speed up the return of the first solution. In any case, using both valid inequalities proves to have a balanced performance between the quality of the solution, its best bound, and the computation time to obtain the first feasible solution.

It is also worth analyzing the effectiveness of the presolve phase carried out by the solver. In Fig. 9 we plot the cumulative distributions of both the root relaxation produced by setting the presolve option to the default and solving the linear relaxation of the models without a presolve phase. It can be observed that although the presolve phase increased the performance of all the models (indeed, for both configurations (2) and (2)–(3) the number of times they reached the highest relaxation value increased of 1), it is especially effective for configuration (3). The increase has such an extent that it went from being dominated by the other two to almost dominating them both. Fig. 9 also shows that without performing this presolve phase, the use of both sets of valid inequalities (2) and (3) had the best performance. This evidence remarks the positive effect of the proposed inequalities, which may be even greater in those contexts where the solver cannot rely on such an effective presolve stage.

7. Conclusions

This study faced a complex staff scheduling problem arising at a major international airport in Northern Italy and involving ground staff. Specifically, we addressed the integrated task scheduling and personnel rostering problem, jointly coping with two problems that are commonly solved in cascade. Inspired by an integrated model from the literature, we proposed two families of valid inequalities.

Computational results showed that jointly solving the task scheduling and the staff rostering problems can be particularly challenging at least for instances coming from the case-study addressed. Even

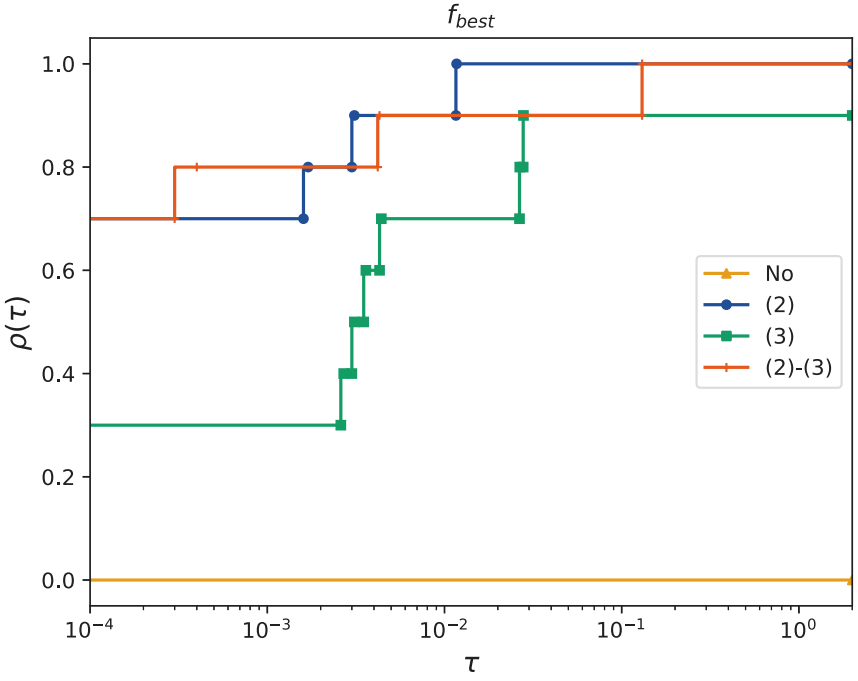

Fig. 6. Cumulative distribution of the relative gap for the best objective value (f_{best}) found by each method.

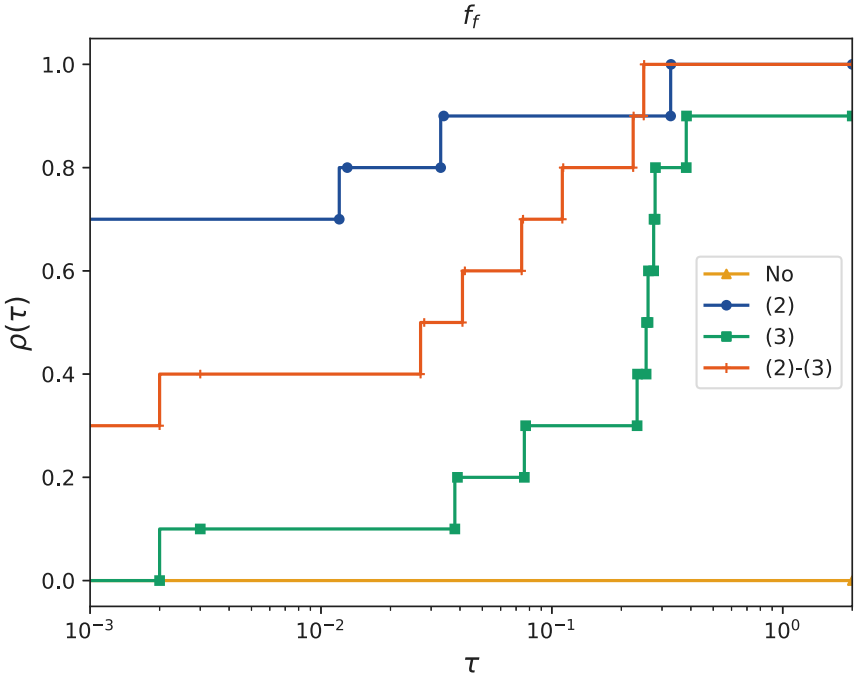

Fig. 7. Cumulative distribution of the relative gap for the objective values at the first feasible solution encountered (f_f) found by each method.

finding feasible solutions becomes critical if the model is not equipped with valid inequalities that significantly improve the performance of the solver used. Indeed, a basic model run without the addition of any valid inequalities fails to provide a feasible solution within the time limit imposed (about 28 h) for all 10 instances. On the contrary, introducing inequalities (2) and (3) may be crucial for finding high-quality solutions in reasonable computation times. Summing up, configuration (2) proved to be a good choice when interested in high-quality solutions; configuration (3) instead proved to be able to find good feasible solutions in the shortest time and a narrow gap to the optimum value; configuration (2)–(3) is a reasonable middle ground between them, able to balance the quality of the best solution and the computation time to get a feasible solution.

We conclude by briefly outlining what future lines of research may be. The proposed approaches have proven successful in solving medium to large instances of the problem, but as the number of requests and employees increases, their performance may degrade. To evaluate the robustness of our methodology, experiments on different problem benchmarks would certainly be of interest. Moreover, it will be interesting to investigate decomposition methods, as well as alternative

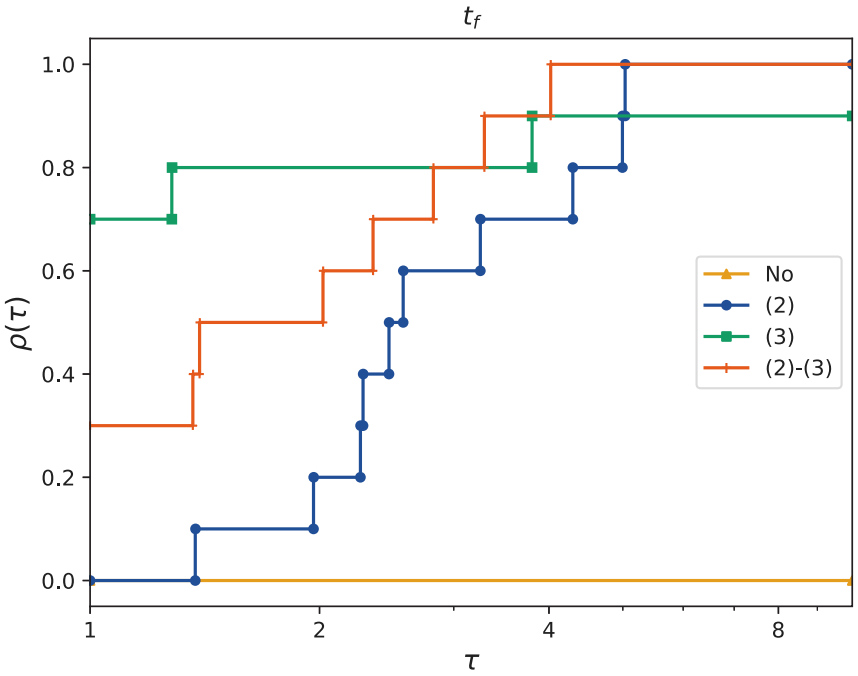

Fig. 8. Performance profiles of the time to retrieve a feasible solution (t_f) required by each method.

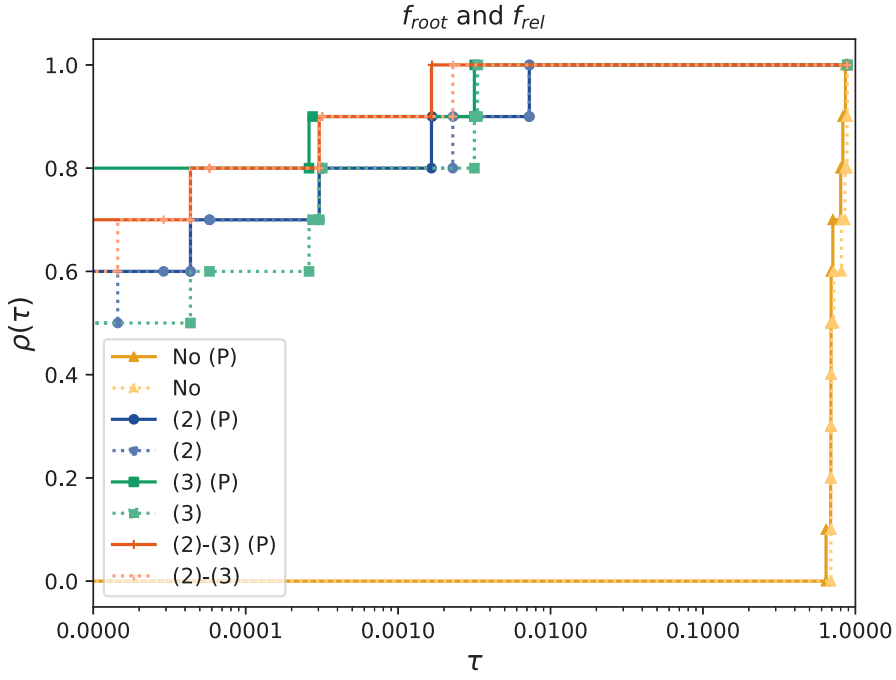

Fig. 9. Cumulative distributions of the relative gap from best objective values; for each model we consider both the root relaxation value after presolve (f_{root}) and the optimal value of pure continuous relaxation (f_{rel}). Note that the reference value for each problem is the overall best, so that we can observe the impact of the presolve operations in increasing the root relaxation bound.

formulations of the problem, e.g., network-based formulations. Furthermore, in such a complex environment as airports, it is crucial to be able to deal with disruptions that may occur as a consequence of flight delays or other events that make planned solutions infeasible. Preliminary results have shown that the proposed approaches are particularly efficient if, as a result of parameter changes, it becomes necessary to reassign tasks to employees on a given day while maintaining planned assignments on other days of the time horizon. Disruption management surely identifies an interesting future development and deserves further study.

CRediT authorship contribution statement

Paola Cappanera: Conceptualization, Methodology, Writing – original draft, Writing – review & editing. **Leonardo Di Gangi:** Data curation, Investigation. **Matteo Lapucci:** Conceptualization, Formal

analysis, Visualization, Writing – original draft, Writing – review & editing. **Giulia Pellegrini:** Data curation, Software. **Marco Roma:** Investigation, Visualization. **Fabio Schoen:** Conceptualization, Methodology, Funding acquisition, Project administration. **Alessio Sortino:** Investigation, Software, Validation.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Fabio Schoen reports financial support was provided by Tuscany Region.

Data availability

The data that has been used is confidential.

Acknowledgments

We would like to thank the people from FirLab s.r.l., and in particular Stefano Cappugi, Massimo Monsignori, and Riccardo Bisori, for the support in working on this project and for sharing the data used for the experiments reported in this manuscript.

Funding

This work was partially supported by the funds from project AIMS - "Artificial Intelligence for the Management of Shifts" - Por Fesr 2014–2020 della Regione Toscana 2014–2020, (azione 1.1.5 sub A1) - FSC 2014–2020 (ex Del.CIPE 40/2020) Progetto "Aiuti agli investimenti R&S delle imprese" - decreto 7056 del 21/04/2021 - CUP D14E20006470009.

Appendix. Finding maximal cliques of incompatible activities

Following the ideas by Krishnamoorthy et al. (2012), to efficiently find all maximal cliques of incompatible tasks, Algorithm 1 can be employed, which exploits the temporal ordering information of requests.

Algorithm 1 Incompatible Cliques of Requests Identification

Require: R set of requests
$T = \emptyset$
for $r \in R$ **do**
  Let (b_r, e_r) the starting and ending times of request r
  $T = T \cup \{(b_r, r)\} \cup \{(e_r, r)\}$
end for
Cast the set of pairs T into a list non-decreasingly order based on the first element of each item (order by times).
$C = \emptyset$
$c = \emptyset$
for $i = 1, \dots, 2|R|$: **do**
  $(t, r) = T[i]$
  if $t == b_r$ **then**
    $c = c \cup \{r\}$
  else
    $C = C \cup c$
    $c = c \setminus \{r\}$
  end if
end for
return C

The algorithm performs the following instructions: first, a sorted list T is created, in non-decreasing order, of all beginning and ending times of all requests. In the list, the identifier r of the corresponding task is also stored. The set of identified cliques C and a "running" clique set c are initialized as empty. Then, the list is iteratively examined; if the current value of t is a starting time, the corresponding request r is added to the running clique c. Otherwise, c is established as a clique, being added to the set of cliques C, and then the request r is removed from it.

We can immediately realize that, at any time, the running clique c indeed contains incompatible requests: all jobs in c have started at a time before t and are finishing at a moment after t. It is also easy to figure out that all maximal cliques are detected by the algorithm. In particular, a maximal clique of incompatible requests is added to C as soon as t represents the end time of the request of the clique that finishes earliest. At that moment, all the requests of the clique have already started and thus have been inserted (and not yet removed) into the running clique c.

We shall also note that, for the algorithm to be correct, if two requests r_1 and r_2 are such that $b_{r_1} = e_{r_2}$, then the pair (b_{r_1}, r_1) should appear later in the ordered list than the pair (e_{r_2}, r_2). Otherwise, r_1 and r_2, that are not incompatible, would be placed together into an incompatibility clique.

Finally, we shall note that the algorithm is computationally cheap: running time is dominated by the sorting operation, thus the computational complexity is $\mathcal{O}(|R| \log(|R|))$.

References

Baatar, D., Krishnamoorthy, M., & Ernst, A. T. (2015). A triplet-based exact method for the shift minimisation personnel task scheduling problem. In N. Bansal, & I. Finocchi (Eds.), *Algorithms-ESA 2015, Lecture Notes in Computer Science, Vol. 9294* (pp. 59–70). Berlin, Heidelberg: Springer.

Brucker, P., Qu, R., & Burke, E. (2011). Personnel scheduling: Models and complexity. *European Journal of Operational Research*, *210*(3), 467–473.

Brusco, M. J., Jacobs, L. W., Bongiorno, R. J., Lyons, D. V., & Tang, B. (1995). Improving personnel scheduling at airline stations. *Operations Research*, *43*(5), 741–751.

Čalnerytė, D., Kriščiūnas, A., & Barauskas, R. (2020). Genetic optimization approach to construct schedule for service staff. In A. Lopata, R. Butkienė, D. Gudonienė, & V. Sukackė (Eds.), *Information and software technologies* (pp. 129–139). Cham: Springer International Publishing.

Campana, N. P., Zucchi, G., Iori, M., Magni, C. A., & Subramanian, A. (2021). An integrated task and personnel scheduling problem to optimize distributed services in hospitals. In *Proceedings of the 23th international conference on enterprise information systems (ICEIS 2021), Vol. 1* (pp. 461–470).

Cappanera, P., & Gallo, G. (2004). A multicommodity flow approach to the crew rostering problem. *Operations Research*, *52*(4), 583–596.

Cappanera, P., Visintin, F., & Rossi, R. (2022). The emergency department physician rostering problem: obtaining equitable solutions via network optimization. *Flexible Services and Manufacturing Journal*, *34*(4), 916–959.

Clausen, T., & Pisinger, D. (2011). *Airport ground staff scheduling*. DTU Management Engineering.

De Bruecker, P., Van den Bergh, J., Beliën, J., & Demeulemeester, E. (2015). Workforce planning incorporating skills: State of the art. *European Journal of Operational Research*, *243*(1), 1–16.

Deveci, M., & Demirel, N. Ç. (2018). A survey of the literature on airline crew scheduling. *Engineering Applications of Artificial Intelligence*, *74*, 54–69.

Dolan, E. D., & Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, *91*, 201–213.

Dowling, D., Krishnamoorthy, M., Mackenzie, H., & Sier, D. (1997). Staff rostering at a large international airport. *Annals of Operations Research*, *72*, 125–147.

Ernst, A. T., Jiang, H., Krishnamoorthy, M., & Sier, D. (2004). Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, *153*(1), 3–27.

Herbers, J., & Hromkovic, J. (2005). *Models and algorithms for ground staff scheduling on airports*: *Technical report*, Fakultät für Mathematik, Informatik und Naturwissenschaften.

Kasirzadeh, A., Saddoune, M., & Soumis, F. (2017). Airline crew scheduling: models, algorithms, and data sets. *EURO Journal on Transportation and Logistics*, *6*(2), 111–137.

Kletzander, L., & Musliu, N. (2020). Solving the general employee scheduling problem. *Computers & Operations Research*, *113*, Article 104794. http://dx.doi.org/10.1016/j.cor.2019.104794, URL: https://www.sciencedirect.com/science/article/pii/S0305054819302369.

Kolen, A. W., Lenstra, J. K., Papadimitriou, C. H., & Spieksma, F. C. (2007). Interval scheduling: A survey. *Naval Research Logistics*, *54*(5), 530–543.

Krishnamoorthy, M., Ernst, A. T., & Baatar, D. (2012). Algorithms for large scale shift minimisation personnel task scheduling problems. *European Journal of Operational Research*, *219*(1), 34–48.

Maheut, J., Garcia-Sabater, J. P., Garcia-Sabater, J. J., & Garcia-Manglano, S. (2023). Solving the multisite staff planning and scheduling problem in a sheltered employment centre that employs workers with intellectual disabilities by MILP: a Spanish case study. *Central European Journal of Operations Research*, URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85161973472&doi=10.1007%2fs10100-023-00864-2&partnerID=40&md5=b37211579522878cb378ea3777d8eb2a.

Mansini, R., Zanella, M., & Zanotti, R. (2023). Optimizing a complex multi-objective personnel scheduling problem jointly complying with requests from customers and staff. *Omega*, *114*, Article 102722. http://dx.doi.org/10.1016/j.omega.2022.102722, URL: https://www.sciencedirect.com/science/article/pii/S0305048322001293.

Meisels, A., & Schaerf, A. (2003). Modelling and solving employee timetabling problems. *Annals of Mathematics and Artificial Intelligence*, *39*(1), 41–59.

Moosavi, A., Ozturk, O., & Patrick, J. (2022). Staff scheduling for residential care under pandemic conditions: The case of COVID-19. *Omega*, *112*, Article 102671. http://dx.doi.org/10.1016/j.omega.2022.102671, URL: https://www.sciencedirect.com/science/article/pii/S0305048322000780.

Nobert, Y., & Roy, J. (1998). Freight handling personnel scheduling at air cargo terminals. *Transportation Science*, *32*(3), 295–301.

Özder, E. H., Özcan, E., & Eren, T. (2020). A systematic literature review for personnel scheduling problems. *International Journal of Information Technology and Decision Making*, *19*(06), 1695–1735.

Rong, A., & Grunow, M. (2009). Shift designs for freight handling personnel at air cargo terminals. *Transportation Research Part E: Logistics and Transportation Review*, *45*(5), 725–739.

Smet, P., Ernst, A. T., & Berghe, G. V. (2016). Heuristic decomposition approaches for an integrated task scheduling and personnel rostering problem. *Computers & Operations Research*, *76*, 60–72.

Smet, P., Wauters, T., Mihaylov, M., & Berghe, G. V. (2014). The shift minimisation personnel task scheduling problem: A new hybrid approach and computational insights. *Omega*, *46*, 64–73.

Van den Bergh, J., Beliën, J., De Bruecker, P., Demeulemeester, E., & De Boeck, L. (2013). Personnel scheduling: A literature review. *European Journal of Operational Research*, *226*(3), 367–385.

Wang, W., Xie, K., Guo, S., Li, W., Xiao, F., & Liang, Z. (2023). A shift-based model to solve the integrated staff rostering and task assignment problem with real-world requirements. *European Journal of Operational Research*, *310*(1), 360–378.