Università di Pisa

Dipartimento di Informatica

Technical Report: TR-01-08

On the Airline Crew Rostering Problem

Paola Cappanera Giorgio Gallo

Revised January 2003

ADDRESS: Corso Italia 40, 56125 Pisa, Italy. TEL: +39 050 2212700 FAX: +39 050 2212726

On the Airline Crew Rostering Problem

Paola Cappanera* Giorgio Gallo

Revised January 2003

Abstract

The problem of finding a work assignment for airline crew members in a given time horizon, is addressed. In the literature such a problem is usually referred to as the *Airline Crew Rostering* problem and consists of constructing monthly schedules for crew members by assigning them pairings, rest periods, leaves, training periods, union activities, and so forth, so as to satisfy the collective agreements and security rules.

We formulate the Airline Crew Rostering problem as a 0-1 *Multicommodity Flow* problem where each employee corresponds to a commodity; determining a monthly schedule for an employee is the same as computing a path on a suitably defined graph while satisfying the clauses of the union conventions.

A preprocessing phase is performed which reduces the dimension of the graph. In order to tighten the linear programming formulation of our model we propose some families of valid inequalities which have proven to be computationally effective; some of them have the characteristic that can be treated implicitly in the construction of the graph.

*email: cappaner@di.unipi.it

Computational results obtained with a commercial integer programming solver (CPLEX) are analyzed.

Keywords. Transportation, scheduling, personnel: crew rostering and Programming, integer, cutting plane/facet: valid inequalities.

1 Introduction

The problem of finding a work assignment for crew members in a given time horizon is addressed. When dealing with the staff management in the airline industry, two types of scheduling problems are considered: the well known *Crew Pairing Problem* and the *Monthly Crew Assignment Problem.* Generally such problems are solved sequentially: first a minimum cost set of *pairings* satisfying the service requirements is constructed; then, in a second phase, monthly schedules for crew members are constructed by assigning them pairings, *reserve blocks* (time periods during which the employee must be available to replace another crew member who cannot work his/her assigned pairing), rest periods, annual leaves, training periods, union activities, and so forth. Each employee must be assigned activities covering all the days of the given time horizon. In both problems collective agreements and security rules have to be guaranteed.

A *pairing* is a trip composed of consecutive legal work assignments, possibly separated by rest periods, to be assigned to a crew leaving and returning to a given base. There are at least three methods for constructing monthly schedules: *bidline*, *rostering* and *preferential bidding.* In a Bidline approach, schedules are first constructed, then their assignment is left to the employees themselves on a seniority basis: this is the case for instance in most North American Companies. The Rostering method [2][13][4], consists in the construction of personalized schedules taking into account a list of preassigned activities; usually an attempt to evenly distribute the workload is made. In the Preferential Bidding approach [11][1][5], schedules are constructed which take into account preassigned activities and a set of employees' weighted bids that reflect their preferences.

This paper describes some aspects of the airline Rostering Problem. The real instances we perform our experiments on are from a medium size Italian airline but the results obtained are quite general: in fact, even though rostering problems are different from airline to airline with respect to rules and costs, they all have the same peculiarities. Relatively similar characteristics can be retrieved also in crew rostering problems at railways and extraurban transportation companies (see [9]), up to the extent that our approach could be suitable also in these setting.

A set of pairings covering the service requirements is given, i.e. we assume that the Crew Pairing Problem has been solved.

The rostering problem is known to be complex and difficult, and this explains why most approaches proposed in the literature are based on heuristics. Usually the rostering problem gives rise to large-scale instances and obtaining good solutions in reasonable time is the main concern. The main ideas underlying the heuristic approaches proposed are the assignment of high priority activities to high priority employees [10][7], the day-by-day assignment of pairings to employees selected from a pool of available crew members [14] and the sequential construction of rosters, employee by employee [11][1]; also combinations of these approaches have been developed [6]. Each of the above methods suffers from several disadvantages, the main one being that they all lack of a global view of the problem and are usually based on the notion of priority which is difficult to ascertain and varies from one airline to another. On the other hand, approaches which take more advantage of the structure of the problem, have been proposed: they are based on a generalized set partitioning model. To solve it, two main methods have been presented: the first one consists of generating a priori a set of feasible rosters for each employee, and solving the resulting problem by integer programming techniques [13]; the other one is a column generation approach [3]. The performance of the first method strongly depend on the goodness of the rosters selected, while a standard column generation approach usually is not able to tackle large scale problems. However in [4] interesting algorithmic control strategies have been proposed and tested to accelerate the solution time of a column generation approach.

On the other hand the size of the real instances solved, which is not too large, has allowed us to tackle the problem via an exact approach and to focus the attention on its structure by exploiting some of its peculiarities. Here the *Airline Crew Rostering* problem, in the following referred to as *ACR*, is formulated as a 0-1 *Multicommodity Flow* problem with additional constraints, where each employee corresponds to a commodity; determining a monthly schedule for an employee is the same as computing a path on a particular graph while satisfying the clauses of the union conventions. Each path on the graph alternates between *activity arcs* representing the activities to be covered in the programming period, and *compatibility arcs* linking together pairs of activities which can be assigned consecutively to the same employee. There are constraints limiting the maximum number of flight and service hours an employee can do, constraints on the maximum number of working days between two rest periods, monthly rest periods constraints etc. These are typical rules and regulations which do not depend on the particular case study (see [9] for a comprehensive description of real world airline crew rostering problems): for instance the way flight and service hours are accounted for may differ from one airline to another but the structure of such constraints is the same for most airline companies.

In this setting the objective of the rostering problem consists of minimizing the number of non-covered activities; however the objective function generally varies from one airline to another and often includes terms which deal with the even distribution of the workload among the crew members. Workload can be expressed in terms of number of monthly rest periods, number of reserve blocks, flight hours, service hours and other quantitative indicators. However throughout the years, equidistribution concerns have been inserted directly into the collective agreements and are thus treated as constraints: this is the reason why here only the minimization of non-covered activities is taken into account in the objective function.

The paper is organized as follows: in Section 2 a detailed description of the rostering problem is presented and a 0-1 Multicommodity Flow model with additional constraints is proposed. In Section 3 some families of valid inequalities are introduced and their properties discussed, in order to tighten the linear programming formulation of the model provided; in Section 4 the valid inequalities proposed are strengthened. Finally in Section 5 computational results are reviewed and some conclusions are drawn.

The main contributions of this work are in the model presented and in the analysis of some aspects of its polyhedral structure, leading to the definition of a family of valid inequalities which are shown to be rather effective from a computational point of view.

2 The model

Given an m days time horizon, a set of n employees and a set of activities to be assigned in the current programming period, one has to find a set of n rosters, one for each employee. In our case the rostering problem deals with three kinds of crew members belonging to the same base, i.e. captains, pilots, and stewards. These three groups of employees represent three independent problems.

ACR can be formulated on a directed graph $G = (N, E)$ where N is the set of nodes and E is the set of arcs. G is an m-layers temporal graph, with a layer for each day t of the current programming period (usually a month). Nodes of level t correspond to beginning and ending times of activities starting or ending in day t. Hereafter we denote by τ_i the time associated with $i \in N$, within the day it belongs to (i.e. $\tau_i \in [0.00, 23.59]$).

For each employee h, also an origin node o^h and a destination node d^h are given.

Let us then introduce the following variables:

x_e^h is the flow along arc $e \in E$ relative to commodity (employee) h,

with $e = (i, j)$, where i is the beginning node of e and j is the ending node of e.

As far as the arcs are concerned, E is composed by the following sets:

- A, the *activity arc set*. An activity can be a pairing, a reserve block, a physiologic rest duty (see later), a day off or a preassigned activity (such as for instance a union activity, a training period, an annual leave or a sick leave). Each activity a is represented in G by an activity arc $e \in A$ between two nodes corresponding to the initial time and the ending time of a respectively. G is an acyclic temporal graph: each $e \in A$ connects a node of layer t with a node belonging to the same or a forward layer, and the numbers of layers involved is just the number of days covered by activity e. Note that an activity arc between two nodes of the same layer corresponds to an activity starting and ending in the same day: this is the case for instance of a day off. A subset of A should be considered apart: R, the *rest arc set* containing only arcs corresponding to rest activities such as day off, sick leave and annual leave.
- C, the *compatibility arc set*, linking the ending node of an activity at layer t with the initial node of a compatible activity at layer $t+1$: as a matter of fact, each activity is long enough to cover at least one day.

More formally stated

$$E = A \cup C,$$

with $R \subseteq A$.

Thus, for each employee h, a roster is just a path on G from the origin o^h to the destination d^h, alternating between activity arcs of A and compatibility arcs of C.

Let us thus introduce for each node $i \in N$

$F_s(i)$	the *forward star* of i which identifies the set of arcs outgoing from i and
$B_s(i)$	the *backward star* of i which is the set of arcs entering into i.

More formally we have:

- if i is the beginning of an activity $e \in A$, then $F_s(i) = \{e\}$

- if i is the ending of an activity $e \in A$, then

  $F_s(i) = \{(i,j) \in C$ s.t. j is the beginning of an activity compatible with $e\}$.

$B_s(i)$ is defined in a similar way.

Activities assigned to h in the current programming period have to link correctly to the activities carried out by h in the previous programming period. For instance in our case, a union clause imposes that there should be at most p working days between two rest periods: so for each employee h, the activities performed in the last p days of the previous programming period must be known. For this purpose the graph G above described is *extended to the left*, i.e. p layers corresponding to the last p days of the previous programming period are added to G and they become the first layers of G in chronological order.

So in G the first p layers correspond to the last p days of the previous programming period; these layers are followed by m layers, one for each day of the current programming period.

Let us analyze more closely how the crossing between two programming periods at the end is made.

In G the correct crossing with the previous programming period is realized by connecting, for each employee h, node o^h to the node which corresponds to the initial time of the first relevant activity assigned to the employee h in the last p days of the previous programming period. Determining a monthly schedule for h is the same as computing a path on G from o^h to d^h where the flow along the arcs belonging to the first p layers of such a path is fixed since the activities covered by h in the previous programming period are known.

On the other hand, if the activities performed by h in the last p days of the previous programming period are unknown (for example because employee h has been recently engaged), then o^h is connected to all the activities which can be performed by h on the first day of the current programming period.

Moreover employee h can be assigned an activity crossing the current programming period and the next one. Without loss of generality, we assume that such activities end on the last day of the current programming period; then in G all the nodes belonging to the last layer and corresponding to the ending time of some activity, are linked to the destination node d^h, for each employee h.

Some information concerning activities and employees should be known in order to formulate the union clauses and collective agreements constraints properly.

Each activity arc $e \in A$ is characterized by the following set of main attributes:

l_e	the *length* of activity e, expressed in number of working days
r_{ek}	the quantity of *resource* k consumed by e
c_e	the number of *crew members* required by e (equal to one for pairings, reserve blocks and preassigned activities, in our case)
Q_e	the set of qualifications required by e
F_e	the set of activities which cannot be performed immediately after e (forbidden pairings).

There is a number of restrictions which cannot be considered implicitly in the construction of graph G since they involve all the arcs in a path. We treat such restrictions as resource constraints by associating each one with a resource k which is accumulated along the path (see r_{ek}) and for which an upper bound r_k is given. There are restrictions on the maximum *flight* and *service time* an employee can collect in each calendar week of the programming period and in the entire month, and restrictions on the minimum number of monthly rest periods (equal to 9 in our case study).

The *flight time* of an activity is the time spent by an employee on board the airplane while performing his/her duties. An activity may include a *must-go*, i.e. a transfer flight from the destination airport of a flight leg to the departure airport of the next leg or a flight from/to the base of the company: the time an employee spends on a transfer flight has to be counted as *service time* and not as flight time. The service time of an activity is a measure of its length and includes in addition to the flight time, times such as the briefing and debriefing time of a pairing, the time spent during a must-go flight, the time spent on union meetings and the training periods.

The set of qualifications required by e, Q_e, depends on the type of aircraft used and on the involved airports: there are for example, some airports, located in critical regions such as near a built-up area or nearby a natural barrier, which require particular skills to land at and take off from.

Moreover for each employee h the following attributes are known:

F^h	the set of activities which cannot be performed by h (forbidden assignments)
Q^h	the set of qualifications of h.

Let us now go into details concerning the way the compatibility arcs set is constructed: usually the linking or compatibility arcs depend on the employee h to whom activities are to be assigned. Anyway there must be a minimum number of hours τ_r between two working days, namely the *physiologic rest period.* As an example, in our case, the physiologic rest period is the maximum between 13 hours and twice the number of flight hours worked in the last working day.

So given the final node i of an activity arc $e \in A$ and an employee h, the forward star of i, $F_s(i)$ can be constructed. $F_s(i)$ is composed by all arcs (i, j), where j is the beginning node of an activity e', such that the following conditions are satisfied: (i) $\tau_j + 24 \geq \tau_i + \tau_r$; (ii) $e' \notin F_e$; (iii) $Q_{e'} \subseteq Q^h$; (iv) $e' \notin F^h$.

Condition (i) is the *physiologic rest period constraint:* it asserts that there must be a rest of at least τ_r hours between the end of activity e and the beginning of activity e'. Condition (ii) imposes that activity e' be compatible with activity e. Condition (iii) asserts that the qualifications of h must comply with the ones required by activity e'; as far as the airport qualifications are concerned, h must be qualified to land and take off in all the airports of pairing e except the ones visited by h as a passenger (a flight leg of a pairing can be a must-go). Finally condition (iv) states that h can be assigned activity e', i.e. the assignment is not a forbidden one.

Observe that the first two conditions are *global constraints* which have to be satisfied for all employees h; on the other hand, conditions (iii) and (iv) are *individual constraints* depending on employee h.

Furthermore when no activity exists which is compatible with e (this happens for example when e is a pairing terminating very late in the night), activity e is linked to the *physiologic rest duty*: when this happens $F_s(i)$ contains only a compatibility arc, the one connecting node i with the initial node of a physiologic rest duty. The physiologic rest duty is a particular rest period to be counted as a working day. In order to avoid misunderstandings we point out here that the physiologic rest period and the physiologic rest duty have two different meanings: the first one is the minimum number of hours elapsed between two consecutive work days, while the latter one is a particular duty in which an employee takes a rest since he/she has finished his/her shift very late in the night, but it is a work day and it must not be counted as a rest day.

Note that when formulating ACR, all above constraints can be treated implicitly in the construction of graph G.

Both the rest day and the physiologic rest duty are activities which can be assigned to any number of employees every day of the programming period; thus in G, for each of them and for each layer, there exists an arc with infinite capacity (i.e. $c_e = \infty$ for each arc e corresponding to a physiologic rest duty or to a rest day).

The construction of personalized schedules must take into account a set of preassigned activities such as annual leaves and training periods. Observe that also the activities covered by an employee in the last p days of the previous programming period are to be considered as preassigned activities. Let thus

P^h be the set of activities preassigned to employee h.

Let us define (or remind) the following parameters

p the maximum number of working days between two rest activities

$d_{\max}$ the maximum length of an activity.

Values for above parameters are given in the regulations of the airline.

Let us then define:

S_t as the set of arcs relative to rest activities which cover layers $t, t+1, \ldots, t+p$

t^h as the day following the last rest day assigned to h in the previous programming period.

ACR can be formulated as

$$
\begin{array}{lllll}
(ACR) & \min \displaystyle\sum_{e \in E} \gamma_e \sum_h x_e^h & & & \\
& \text{s.t.} \displaystyle\sum_{e \in F_s(o^h)} x_e^h = 1 & \forall h & & (1) \\
& \displaystyle\sum_{e \in B_s(d^h)} x_e^h = 1 & \forall h & & (2) \\
& \displaystyle\sum_{e \in B_s(i)} x_e^h - \sum_{e \in F_s(i)} x_e^h = 0 & \forall h & \forall i \in N & (3) \\
& \displaystyle\sum_h x_e^h \leq c_e & & \forall e \in A & (4) \\
& \displaystyle\sum_{e \in S_t} x_e^h \geq 1 & \forall h & \forall t \in \{t^h, t^h+1, \ldots, m\} & (5) \\
& \displaystyle\sum_{e \in A} r_{ek} x_e^h \leq r_k & \forall h & \forall k & (6) \\
& x_e^h = 1 & \forall h & \forall e \in P^h & (7) \\
& x_e^h = 0 & \forall h & \forall e \in F^h & (8) \\
& x_e^h \in \{0,1\} & \forall h & \forall e \in E & (9)
\end{array}
$$

where the costs γ_e's are defined either as:

$$\gamma_e = \begin{cases} -1 & \text{if } c_e < +\infty \\ 0 & \text{otherwise,} \end{cases}$$

or as

$$\gamma_e = \begin{cases} -l_e & \text{if } c_e < +\infty \\ 0 & \text{otherwise.} \end{cases}$$

In the first case, referred to as the *max cardinality* case, the objective function consists of maximizing the number of covered activities, while in the second case (referred to as *max length* case) one wants to maximize the total length of activities covered, so as to minimize the duration of uncovered activities which will be assigned to extra staff in order to guarantee the service requirements.

ACR thus consists of maximizing either the number or the total length of covered activities while satisfying a set of constraints.

In the following, a description of each type of constraints is presented.

Flow conservation constraints.

A monthly roster for an employee h is a path on G from origin o^h to destination d^h. Constraints (1) (2) and (3) thus state that for each employee h a unitary flow goes out of the source o^h and enters into the sink d^h while any other node of G is a transshipment node, i.e. it has a null demand.

The correct linking of the current programming period with the previous and the next ones, as well as several compatibility constraints between two consecutive activities are implicitly treated in the structure of graph G.

Mutual capacity constraints.

Constraints (4) limit the total quantity of flow on activity arc $e \in A$, regardless of the commodity. They assure that each activity $e \in A$ is assigned to at most c_e employees, with $c_e = +\infty$ for each arc corresponding to a rest day or to a physiologic rest duty, and equal to one for pairings, reserves and preassigned activities.

Working days constraints.

These constraints assure that for each employee h the number of consecutive working days between rest periods is at most p ($p = 6$ in our case study), i.e. every $p+1$ days, h must be assigned at least one rest activity.

Constraints (5) thus guarantee that, for each employee h and for each day t, the number of rest periods assigned to h in the time window $[t, t+p]$ is at least equal to one, provided that the interval $[t, t+p]$ ends in the current programming period (i.e. $\forall t$ s.t. $t \leq m$ since the in G there are $m+p$ layers).

Note that for each employee h, the rest periods actually assigned to him/her in the last days of the previous programming period are taken into account, since graph G has been extended to the left.

Moreover, observe that, for each employee h, the actual number of constraints (5) depends on the last rest period assigned to h in the previous programming period. For example if employee h has not been assigned a rest period in the last p days of the previous programming period, then he/she must be assigned a rest period in the first day of the current programming period.

The correct number of rest activities in the entire programming period is then assured.

Here only daily rest periods are assumed. Longer leaves are represented as sequences of daily periods: as an example a 15 days' leave is represented by 15 consecutive rest periods of length equal to one.

Resource constraints.

Constraints (6) are the resource constraints which, employee by employee, assure that the level of consumption of each resource k along a path must not exceed the threshold r_k. Examples of resources are weekly and monthly flight and service time, number of rest periods, reserve blocks and so forth.

Preassigned activities constraints.

For each employee h constraints (7) fix to one the flow along all the arcs corresponding to preassigned activities.

Forbidden assignments constraints.

For each employee h constraints (8) fix to zero the flow on all the arcs corresponding to activities which cannot be assigned to h.

3 Valid inequalities

In this section some aspects of the polyhedral structure of ACR are investigated. The polyhedron P defined by constraints (1),..., (8) and by the continuous relaxation of constraints (9), can be described in terms of the intersection of $n+1$ polyhedra: a polyhedron for each commodity $h \in \{1, \ldots, n\}$ plus a polyhedron defined by constraints (4) which link together the otherwise separable single commodity flows. If constraints (4) were not present, ACR would be decomposable in n subproblems, each of them defined by all the constraints relative to a given h. But looking at the very structure of polyhedron (relative to commodity) h, we observe that, at its turn, it can be described in terms of the intersection of the following three polyhedra:

- P_1 the flow polyhedron given by the single commodity flow conservation constraints (1),(2),(3),(7) and (8)
- P_2 the polyhedron given by the resource constraints (6)
- P_3 the polyhedron given by the working day constraints (5) which can be

arranged so as to exhibit a (block) diagonal structure with a bandwidth equal to $p+1$.

In order to tighten the linear programming formulation of ACR, studying the structure of the convex hull of the 0-1 points in $P_1 \cap P_2 \cap P_3$ is thus worth considering. In particular we are going to characterize separately the structure of the convex hulls of the 0-1 points in $P_1 \cap P_2$ and in $P_1 \cap P_3$ which can result in a good approximation of the convex hull of the 0-1 points in $P_1 \cap P_2 \cap P_3$, as the computational results of Section 5 show. In this section some families of valid inequalities for ACR are presented, which are successively shown to be facet inducing for a polyhedron related either to the convex hull of the 0-1 points in $P_1 \cap P_2$ or in $P_1 \cap P_3$, when some assumptions hold.

Let us introduce some definitions: let

$A^h \subseteq A \setminus R$	be the set of all the service activities which can be assigned to employee h.

Definition 1. *A set $C^h_{td} \subseteq A^h$ is called a cover for employee h, if it identifies a subpath on G^h, i.e. a sequence of non-overlapping activities, pair-wise connected by compatibility arcs, which starts at day t and covers d consecutive working days.*

We assume that the first activity of subpath C^h_{td} begins on day t and the last one ends on day $t+d-1$.

We now focus the attention on constraints (5) which assure that for each employee h the number of consecutive working days between two rest periods is at most p.

The inequality

$$\sum_{e \in C^h_{t,p+1}} x^h_e \leq |C^h_{t,p+1}| - 1, \tag{10}$$

is valid for ACR since no feasible solution exists with $x^h_e = 1, \forall e \in C^h_{t,p+1}$: in fact, should such a solution exist, employee h would perform $p+1$ consecutive working

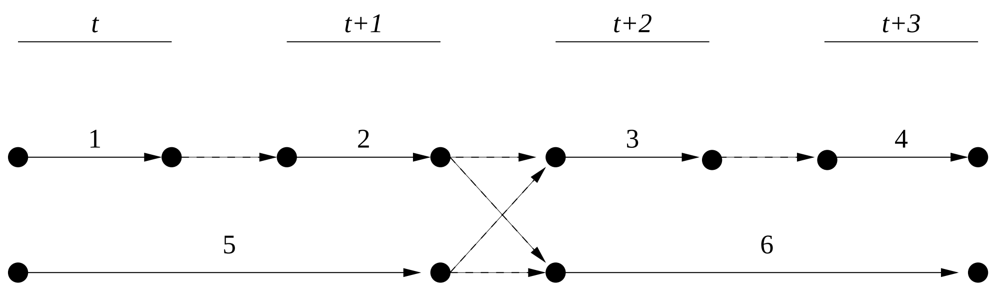

Figure 1: An example of graph involving $p+1$ layers ($p=3$)

days without a rest, thus violating constraints (5).

We now show an example to better understand the notation used: suppose that $p = 3$ and consider a time window of 4 working days starting on day t and ending on day $t+3$. Suppose that for the employee h in the time period considered there are 4 rest activities, namely arcs $1, 2, 3, 4$, one for each day, and 2 service activities, both of them 2 days long, namely arcs 5 and 6. Such a situation is sketched in Figure 1 where dashed lines represent the compatibility arcs. Since there exists a compatibility arc between activities 5 and 6, $C^h_{t4} = \{5, 6\}$ identifies a feasible subpath of length 4, which violates the working day constraint.

In a situation like this

$$x^h_5 + x^h_6 \leq 1,$$

is a valid inequality for ACR, assuring that activities 5 and 6 cannot be both assigned to employee h.

Let us now give some insight on the properties of valid inequality (10).

If $e = (i, j)$ is a service activity arc, with i belonging to day t_1 and j belonging to day t_q, we call $R(e)$ the set of rest arcs corresponding to days $t_1, t_2, \ldots, t_q$.

Given a subpath $C^h_{t,p+1}$, for any two arcs e and k belonging to it, it is true that

$$R(e) \cap R(k) = \emptyset.$$

In fact, belonging both to $C^h_{t,p+1}$, they correspond to services which can be assigned to the same roster and therefore which do not overlap.
Clearly, $\{R(e) : e \in C^h_{t,p+1}\}$, for any h, defines a partition of the arcs in S_t.
Then the following constraints hold

$$\sum_{e \in S_t} x^h_e \geq 1, \tag{5}$$

$$x^h_e + x^h_k \leq 1, \quad \forall k \in R(e), \quad \forall e \in C^h_{t,p+1}, \tag{11}$$

where constraints (11) are implied by the flow conservation constraints relative to commodity h.
Let us define P^h_t as

$$P^h_t = \{x^h_e \text{ s.t. } (5), (11), x^h_e \in \{0,1\}, \forall e \in C^h_{t,p+1} \cup S_t\}.$$

For example, coming back to the situation represented in Figure 1, we have $S_t = \{1,2,3,4\}$, service activity 5 covers days t and $t+1$, so that $R(5) = \{1,2\}$, while service activity 6 covers days $t+2$ and $t+3$ and $R(6) = \{3,4\}$. Clearly $R(5)$, $R(6)$ identify a partition of S_t. The valid inequality above stated, cuts off the following fractional solution

$$x^h_1 = x^h_2 = 0, \quad x^h_3 = x^h_4 = 1/2, \quad x^h_5 = 1, \quad x^h_6 = 1/2,$$

which is feasible to the linear relaxation of P^h_t.
Let us then denote with $\overline{P}^h_t$ the continuous relaxation of P^h_t, with $\dim(\overline{P}^h_t)$ the dimension of polyhedron $\overline{P}^h_t$ and finally with $\mathrm{conv}(P^h_t)$ the convex hull of P^h_t.

Proposition 1. *$\overline{P}_t^h$ is full-dimensional.*

Proof. Let us assume w.l.g. that $|S_t| = p+1$.

It is straightforward to see that point $\overline{x}^h$ defined as follows:

$$\overline{x}_e^h = \begin{cases} 1-\varepsilon & \text{if } e \in S_t \\ \varepsilon/2 & \text{if } e \in C_{t,p+1}^h \end{cases}$$

is an interior point of $\overline{P}_t^h$ for each ε such that $0 < \varepsilon < p/(p+1)$. In fact the box constraints $0 \leq x_e^h \leq 1$ are never satisfied as equality by $\overline{x}_e^h$ since $\varepsilon > 0$; moreover $\overline{x}_e^h$ satisfies constraints (5) and (11) as strict inequality respectively for $\varepsilon < p/(p+1)$ and $\varepsilon > 0$. $\overline{P}_t^h$ is thus full-dimensional and $\dim(\ \overline{P}_t^h\ ) = p + 1 + |C_{t,p+1}^h|$. □

Now we go into details concerning the valid inequalities shown up to now. In particular we address the question whether such inequalities are facet-inducing for $\text{conv}(P_t^h)$ or not.

In the following an activity covering only one day will be referred to as *singleton*.

Lemma 1. *Given a cover $C_{t,p+1}^h$, for each non-singleton activity e in $C_{t,p+1}^h$, the maximum number of linearly independent vectors of P_t^h which satisfies (10) as equality by fixing the corresponding variable x_e^h to zero is equal to $l_e + 1$.*

Proof. First we show a possible way of obtaining a set of $l_e + 1$ vectors, then their linear independence is proven.

Given service activity e in $C_{t,p+1}^h$, $R(e)$, which is the set of rest arcs covering the same layers as e, has cardinality l_e and we denote with r_k the k-th element of $R(e)$, i.e.

$$R(e) = \{r_1, r_2, \ldots, r_{l_e}\}.$$

For a given e we define a set of $l_e + 1$ vectors $v_1^e, v_2^e, \ldots, v_{l_e+1}^e$, which have one component for each $s \in C_{t,p+1}^h$, for each $r \in R(s)$. For $k \in \{1, 2, \ldots, l_e\}$, v_k^e is defined as follows: the components corresponding to r_k and to the elements of $C^h_{t,p+1} \setminus \{e\}$ have value one, while all the other components are value zero. The rationale is that vectors v^e_k are obtained by fixing the variable relative to service activity e to zero. Then all variables relative to service activities but activity e are set to one in order to satisfy (10) as equality. Constraints (11) assure that $x^h_r = 0$, $\forall s \in C^h_{t,p+1} \setminus \{e\}$, $\forall r \in R(s)$. Thus, in order to satisfy constraints (5), we fix to one, only one of the rest variable in $R(e)$: there are l_e ways of choosing such a variable and thus l_e vectors v^e_k are obtained. In particular, in vector v^e_k the rest variable relative to the k-th element of $R(e)$ is fixed to one.

Finally $v^e_{l_e+1}$ is defined as follows: all the components corresponding to elements of $R(e)$ and of $C^h_{t,p+1} \setminus \{e\}$ are fixed to one, while all the other components have value zero.

W.l.g. we can assume that the first $p+1$ components of v^e_k refer to the rest arcs and the last $|C^h_{t,p+1}|$ ones correspond to the service activities belonging to cover $C^h_{t,p+1}$. Then the $l_e + 1$ vectors can be arranged so as to form the following matrix:

$$M_e = \left[\begin{array}{ccc|c} & & & 1 \\ & I_{l_e} & & \cdot \\ & & & 1 \\ \hline 0 & \cdot & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & 0 & 0 \\ \hline 0 & \cdot & \cdot & 0 \\ \hline 1 & \cdot & \cdot & 1 \\ \cdot & \cdot & \cdot & \cdot \\ 1 & \cdot & \cdot & 1 \end{array}\right] \begin{array}{l} \left.\vphantom{\begin{array}{c}1\\ \cdot \\ 1\end{array}}\right\} l_e \\ \left.\vphantom{\begin{array}{c}0\\ \cdot \\ 0\end{array}}\right\} p+1-l_e \\ \vphantom{0} \\ \left.\vphantom{\begin{array}{c}1\\ \cdot \\ 1\end{array}}\right\} |C^h_{t,p+1}|-1 \end{array} .$$

where w.l.g. we assume that activity e is the first activity in the cover. Observe that the first two blocks are relative to the rest variables: the former comprises the l_e rest variables in $R(e)$, while the latter is relative to all of the other rest variables. The third block consists of a single row relative to service activity e which is set to 0; finally the last block is relative to the service activities other than e which are set to 1.

We observe that $\text{rank}(M_e) \leq l_e + 1$ but it is easy to show that $\text{rank}(M_e)$ is equal to $l_e + 1$: as a matter of fact the submatrix of M_e obtained by picking the first l_e rows of M_e and a all-ones row from the last block, i.e. $\left[\begin{smallmatrix} I_{l_e} & 1 \\ 1 & 1 \end{smallmatrix}\right]$ by a proper linear combination of the rows results in the following diagonal matrix with nonzero diagonal elements: $\left[\begin{smallmatrix} I_{l_e} & 1 \\ 0 & l_e - 1 \end{smallmatrix}\right]$.

Finally it is easy to see that any other feasible vector to P_t^h which differs from the columns in M_e only in the first l_e components is linearly dependent from them and this completes the proof. $\square$

Theorem 1. *Given a cover $C_{t,p+1}^h$, let k be the number of singletons. The valid inequality (10) induces a face of conv(P_t^h) of dimension dim($\overline{P}_t^h$) $- k - 1$.*

Proof. As shown in Proposition 1, $\overline{P}_t^h$ is full-dimensional: in order to prove that (10) is a face of $\text{conv}(P_t^h)$ of dimension $\dim(\overline{P}_t^h) - k - 1$, we have to provide $\dim(\overline{P}_t^h) - k$ linearly independent vectors which are feasible to P_t^h and which satisfy (10) as equality. Valid equality (10) is satisfied as equality, if at turn one of the $|C_{t,p+1}^h|$ service activities in $C_{t,p+1}^h$ is set to zero while all the other activities in the cover are set to one. Lemma 1 proves that, for a given non-singleton e in $C_{t,p+1}^h$ we can construct $l_e + 1$ linearly independent vectors with $x_e^h = 0$.

Let us now examine the case of a singleton activity e: if x_e^h is set to 0, then x_j^h is set to one $\forall j \in C_{t,p+1}^h$ with $j \neq e$ in order to satisfy the valid inequality (10) as equality. Consequently the corresponding rest variables are set to zero and the only way to satisfy the working day constraint is to set the rest variable relative to singleton e to one. So we can conclude that for each singleton e we are able to build only one vector with $x_e^h = 0$. Let us now form a matrix whose columns are the vectors described above, i.e. $l_e + 1$ vectors for each non-singleton e and a vector in correspondence to each singleton: the resulting matrix is shown below on the left, while on the right the matrix obtained by linear combinations of its rows is provided. Let q be the number of non-singleton activities in the cover, i.e. $q = |C^h_{t,p+1}| - k$ and let us assume w.l.g. that the elements in the cover are ordered so that the first q ones correspond to non-singleton activities. We also use the notation b_{rc} with $b \in \{0,1\}$ to denote a matrix with r rows and c columns and elements are all equal to b.

$$\left[\begin{array}{c|c} \begin{matrix} I_{l_1} & 1_{l_1} & & \\ & \cdot & & \\ & & I_{l_q} & 1_{l_q} \end{matrix} & 0_{p+1-k\times k} \\ \hline O_{k\times \dim(\overline{P}^h_t)-k} & I_k \\ \hline B & 1_{q\times k} \\ \hline 1_{k\times \dim(\overline{P}^h_t)-k} & 1_{k\times k} - I_k \end{array}\right] \left[\begin{array}{c|c} \begin{matrix} I_{l_1} & 1_{l_1} & & \\ & \cdot & & \\ & & I_{l_q} & 1_{l_q} \end{matrix} & 0_{p+1-k\times k} \\ \hline B & 0_{q\times k} \\ \hline 1_{k\times \dim(\overline{P}^h_t)-k} & 0_{k\times k} \\ \hline 0_{k\times \dim(\overline{P}^h_t)-k} & I_k \end{array}\right].$$

By definition, in the left matrix above, for each singleton e the row (second block) corresponding to the unique rest variable in $R(e)$ contains only a 1 in column relative to e. By subtracting such k rows from all the last $|C^h_{t,p+1}|$ rows in the left matrix and shifting the k rows to the bottom of the matrix, we obtain the matrix on the right. Let us now go into details concerning submatrix B: such a matrix has a row $r(e)$ for each non singleton activity e. Recall the reader that such a block has been constructed by providing for each non-singleton e, $l_e + 1$ vectors with $x^h_e = 0$; so each row $r(e)$ in B is a all-ones row but in positions corresponding to the $l_e + 1$ vectors. Given a non-singleton activity e we can thus substitute row $[r(e), \mathbf{0}]$ with one of the rows in $[1_{k\times \dim(\overline{P}^h_t)-k}, \mathbf{0}]$ minus row $[r(e), \mathbf{0}]$ thus obtaining a all zeros row but in positions corresponding to the $l_e + 1$ vectors where we have a 1. Then we insert the row thus obtained just after the block $[I_{l_e}, 1_{l_e}]$. Repeating such a procedure for each non-singleton e and exploiting the linear combinations used to prove Lemma 1, we obtain the following matrix:

$$
\left.\left[\begin{array}{cccccc|c}
I_{l_1} & 1_{l_1} & & & & & \\
0 & l_1 - 1 & & & & & \\
& & \cdot & & & & 0_{\dim(\overline{P}_t^h)-k\times k} \\
& & & & I_{l_q} & 1_{l_q} & \\
& & & & 0 & l_q - 1 & \\
\hline
\multicolumn{6}{c|}{0_{k\times\dim(\overline{P}_t^h)-k}} & I_k \\
\hline
\multicolumn{6}{c|}{1_{k\times\dim(\overline{P}_t^h)-k}} & 0_{k\times k}
\end{array}\right]\right\} \dim(\overline{P}_t^h)
$$

The submatrix obtained by eliminating the last k rows is a diagonal block matrix with non zero diagonal elements, and this completes the proof. $\square$

Let us now show an example to better understand how the vectors are generated and how their linear independence is proven. For matter of convenience in the following, while showing an application of Theorem 1, we use the name x to identify variables corresponding to rest activity arcs and y to refer to service activity arcs; we then omit the index h which refers to the employee.

Suppose that $p = 5$ and the following constraints hold:

$$
\begin{aligned}
&x_1 + x_2 + x_3 + x_4 + x_5 + x_6 && \geq 1 && (12)\\
&x_1 + y_1 && \leq 1 && (13)\\
&x_2 + y_1 && \leq 1 && (14)\\
&x_3 + y_1 && \leq 1 && (15)\\
&x_4 + y_2 && \leq 1 && (16)\\
&x_5 + y_2 && \leq 1 && (17)\\
&x_6 + y_3 && \leq 1 && (18)
\end{aligned}
$$

where y_1 and y_2 correspond to service activities 3 and 2 days long respectively, and y_3 correspond to a singleton. Let then P be defined as

$$\mathrm{P} = \{(x, y) \text{ s.t. } (12), (13), (14), (15), (16), (17), (18), x \in \{0,1\}^6, y \in \{0,1\}^3\}.$$

Observe that summing up constraints involving both rest variables x and service variables y, and subtracting inequality (12) to the constraint thus obtained, we get the inequality $3y_1 + 2y_2 + y_3 \leq 5$ which is weaker than the valid inequality $y_1 + y_2 + y_3 \leq 2$; such an inequality cuts off for instance the following solution feasible to $\overline{P}$:

$$x_1 = x_2 = x_3 = 2/3, \quad x_4 = x_5 = x_6 = 0, \quad y_1 = 1/3 \quad y_2 = y_3 = 1.$$

By applying the procedure described in Theorem 1, we obtain the following matrix, where 8 vectors are given column by column, as explained before: for matter of convenience beside each row the name of the corresponding variable is reported.

$$\begin{array}{c} (x_1) \\ (x_2) \\ (x_3) \\ (x_4) \\ (x_5) \\ (x_6) \\ (y_1) \\ (y_2) \\ (y_3) \end{array} \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}.$$

For example by fixing $y_1 = 0$ (and consequently $y_2 = y_3 = 1$) we are able to provide 4 (since $l_1 = 3$) linearly independent vectors which are the first four columns of the matrix above. In this example matrix $B = \left[\begin{smallmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{smallmatrix}\right]$ has two rows corresponding to non-singleton activities y_1 and y_2. By subtracting row corresponding to x_6 from the last 3 rows and moving it to the bottom of the matrix we obtain the left matrix below.

$$
\begin{array}{c}
(x_1)\\ (x_2)\\ (x_3)\\ (x_4)\\ (x_5)\\ (y_1)\\ (y_2)\\ (y_3)\\ (x_6)
\end{array}
\begin{bmatrix}
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0\\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0\\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0\\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 0\\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0\\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\qquad
\begin{array}{c}
(x_1)\\ (x_2)\\ (x_3)\\ (y_1)\\ (x_4)\\ (x_5)\\ (y_2)\\ (x_6)\\ (y_3)
\end{array}
\left[\begin{array}{cccc|ccc|c}
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0\\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0\\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0\\
\hline
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0\\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0\\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1\\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
\hline
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0
\end{array}\right].
$$

The right matrix has been obtained applying the following linear operations to the left matrix: $y_1 \leftarrow y_3 - y_1$ and $y_2 \leftarrow y_3 - y_2$. We observe that the submatrix obtained from the right matrix by eliminating the last row exhibits a diagonal block structure with non-zero diagonal elements: its 8 columns are linearly independent and this shows that $y_1 + y_2 + y_3 \leq 2$ is a face of dimension 7.

Corollary 1. *If the number of singletons in cover $C^h_{t,p+1}$ is equal to 0, the valid inequality (10) is a facet for conv(P^h_t).*

Corollary 2. *If $l_e = 1, \forall e \in C^h_{t,p+1}$, then valid inequality (10) is redundant for conv(P^h_t).*

Proof. By summing up constraints (11) we get the inequality

$$\sum_{e \in C^h_{t,p+1}} l_e x^h_e + \sum_{e \in C^h_{t,p+1}} \sum_{k \in R(e)} x^h_k \leq p+1,$$

which is the same as

$$\sum_{e \in C^h_{t,p+1}} l_e x^h_e + \sum_{e \in S_t} x^h_e \leq p+1,$$

since the sets $R(e)$ form a partition of S_t. By subtracting constraint (5) to the inequality just obtained we get

$$\sum_{e \in C^h_{t,p+1}} l_e x^h_e \leq p. \tag{19}$$

Observe that when $l_e = 1, \forall e \in C^h_{t,p+1}$, valid inequality (10) is exactly the same as constraint (19) since in this case $|C^h_{t,p+1}| - 1 = p$. □

Indeed we can strengthen the result above by showing that when the cover is composed only by singletons, constraints defining $\overline{P}^h_t$ fully describe conv(P^h_t).

Corollary 3. *If $l_e = 1, \forall e \in C^h_{t,p+1}$ then $\overline{P}^h_t$ = conv(P^h_t).*

It is in fact straightforward to see that the constraints matrix defining $\overline{P}^h_t$ is totally unimodular.

An alternative way of getting valid inequality (10) from constraint (19) is explained in the following. To this end we collect here some basic results on polyhedra and some standard definitions (for more details the reader is referred to [12]).

Consider a 0-1 knapsack problem and let

$$F = \{x \in \{0,1\}^n : \sum_{j=1}^{n} a_j x_j \leq b\}$$

be its feasible set where $a_j \in Z_+$ for $j \in \{1, \ldots, n\}$ and $b \in Z_+$. Since $a_j > b$ implies $x_j = 0$ for all $x \in F$, we assume $a_j \leq b$ for all j. Thus the dimension of conv(F) is n.

Definition 2. [12] Given a set $C \subseteq \{1, \ldots, n\}$, we say that C is an independent set if $\sum_{j \in C} a_j \leq b$; otherwise C is a dependent set.

Proposition 2. [12] If C is a dependent set, then

$$\sum_{j \in C} x_j \leq |C| - 1,$$

is a valid inequality for F.

We observe that $C^h_{t,p+1}$ is a dependent set since $\sum_{e \in C^h_{t,p+1}} l_e = p+1 > p$ and then constraint (10) is a valid inequality for set $\{\sum_{e \in C^h_{t,p+1}} l_e x^h_e \leq p\}$.
Furthermore we have that $C^h_{t,p+1}$ is a *minimal dependent set.*

Definition 3. [12] A dependent set is minimal if all of its subsets are independent.

As far as set $C^h_{t,p+1}$ is concerned it is sufficient to remove an element from it in order to satisfy constraint (19).

Finally, let $\overline{F}^h_t$ be the polyhedron obtained by adding valid inequality (10) to $\overline{P}^h_t$: we observe that, if an $e \in C^h_{t,p+1}$ exists such that $l_e > 1$, (10) is necessary to give the description of $\overline{F}^h_t$.
$\overline{F}^h_t$ has the same dimension as $\overline{P}^h_t$: in fact for each ε such that $0 < \varepsilon < \min\{p/(p+1), 2(1-1/|C^h_{t,p+1}|)\}$, x^h_e defined in Proposition 1 is an interior point also to $\overline{F}^h_t$. In order to prove that (10) is a facet of $\overline{F}^h_t$ we have to provide dim($\overline{F}^h_t$) - 1 linearly independent vectors which are feasible to $\overline{F}^h_t$ and satisfy (10) as equality. To this end we enlarge the set of vectors given in Theorem 1 by adding one more vector for each singleton activity e, namely vector $v^e_{l_e+1}$, which is defined as follows: the component corresponding to the only rest variable r_1 in $R(e)$ is fixed to zero, while the components relative to all the other p rest variables are fixed to the same value $1/p$. Then the components corresponding to service activities are all fixed to the same value $1-1/|C^h_{t,p+1}|$. It is possible to show that such new vectors are linearly independent from the set of vectors previously defined and the enlarged set of vectors spans $\overline{F}^h_t$. In particular by proper linear combinations of the rows, the matrix, whose columns are the vectors from the enlarged set, can be shown to exhibit a block diagonal structure. In such a matrix there is a block for each service activity in $C^h_{t,p+1}$: an identity block of dimension 2 for each singleton and an upper triangular block of dimension l_e+1 with non-zero diagonal elements, for each service activity covering $l_e > 1$ days.

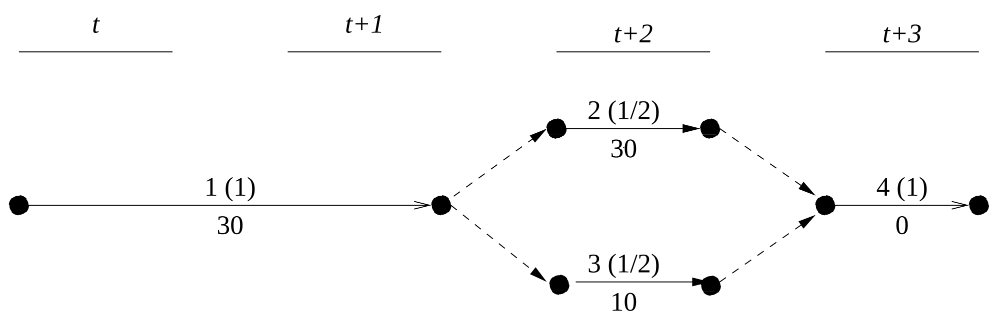

Figure 2: An example involving the flight time

From the non-singularity of the matrix, the linear independence of the vectors follows.

Other valid inequalities can be introduced by taking into account the resource constraints (6). Without loss of generality we focus on the resource constraint relative to the maximum flight time in a week (time period of length 7). Let d be a positive integer number such that $d \leq \min\{7, p\}$, C^h_{td} be defined as in Definition 1 and assume that the feasible subpath C^h_{td} violates the resource constraint. Then valid inequality

$$\sum_{e \in C^h_{td}} x^h_e \leq |C^h_{td}| - 1, \tag{20}$$

holds.

Consider the following example where we have three service activity arcs, namely 1, 2 and 3, and a rest arc, namely 4, as shown in Figure 2. Below each arc the flight time of the corresponding activity arc is reported, and we assume that the maximum number of flight hours an employee can do in a week be equal to 50. Fixed $d = p = 3$, $C^h_{t3} = \{1, 2\}$ is a feasible subpath which violates the resource constraint: in fact $30 + 30 > 50$.

Constraint $x^h_1 + x^h_2 \leq 1$, is thus a valid inequality; for instance it cuts off the solution

$x_1^h = 1$, $x_2^h = x_3^h = 1/2$, which satisfies the resource constraint being $30 + 1/2 \cdot 30 + 1/2 \cdot 10 = 50$.

4 Lifting the valid inequalities

The valid inequality (10) can be further tightened: to this end a little bit more notation has to be introduced.

Given a service activity arc e and a given time period, let us define $A(e)$ as the set of service activity arcs which cover the same layers as e in the time period considered, i.e. $A(e) = \{k \in A \setminus R : k$ covers the same layers of $e\}$. More formally stated, for each service activity e let

t_e^1	be the beginning day of activity e
t_e^q	be the ending day of activity e.

Given a path $C_{t,p+1}^h$ covering $p+1$ layers, for each activity arc e belonging to $C_{t,p+1}^h$, $A(e)$ is thus defined as follows

$$A(e) = \begin{cases} \{k \in A \setminus R \text{ s.t. } l_k \geq l_e,\ t_e^q = t_k^q\} & \text{if } e \text{ is the first activity of subpath } C_{t,p+1}^h \\ \{k \in A \setminus R \text{ s.t. } l_k = l_e,\ t_e^q = t_k^q\} & \text{if } e \text{ is an activity in the middle of} \\ & \text{subpath } C_{t,p+1}^h \\ \{k \in A \setminus R \text{ s.t. } l_k \geq l_e,\ t_e^1 = t_k^1\} & \text{if } e \text{ is the last activity of subpath } C_{t,p+1}^h. \end{cases}$$

Then the following valid inequality holds:

$$\sum_{e \in C_{t,p+1}^h} \left(x_e^h + \sum_{k \in A(e)} x_k^h \right) \leq |C_{t,p+1}^h| - 1, \tag{21}$$

since the constraint

$$x_e^h + x_k^h \leq 1, \quad \forall k \in A(e), \quad \forall e \in C_{t,p+1}^h,$$

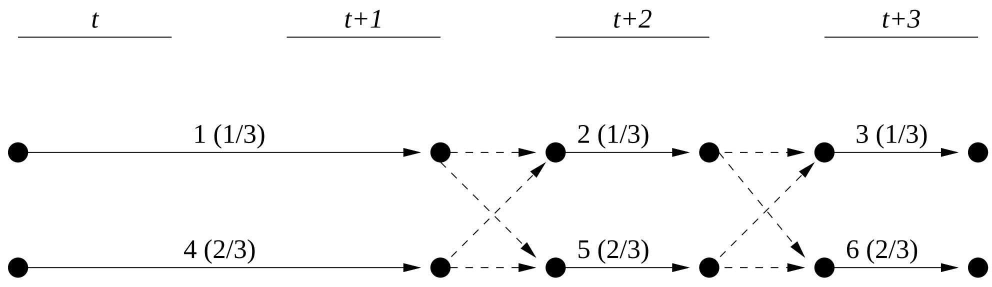

Figure 3: Strengthening the valid inequality (10): an example for $p = 3$

holds, being e and k overlapping activities.

In order to better understand the valid inequality (21) we consider the following example sketched in Figure 3. Here there are two rest arcs, namely 2 and 3; and 4 service activity arcs, i.e. 1,4,5, and 6. In brackets, for each arc, we report the value of a fractional feasible flow: for the employee h we have a subpath composed by arcs 1, 2 and 3 with a flow equal to 1/3 and another subpath given by arcs 4, 5 and 6 with a flow equal to 2/3. We have $C^h_{t,p+1} = \{4, 5, 6\}$, $A(4) = \{1\}$, $A(5) = A(6) = \emptyset$: as a matter of fact activity 1 and 4 have the same length and cover the same time period. The valid inequality $x^h_4 + x^h_5 + x^h_6 \leq 2$, does not cut off the current fractional solution, while the tightened valid inequality $x^h_1 + x^h_4 + x^h_5 + x^h_6 \leq 2$, cuts off the fractional solution being $1/3 + 3 \cdot 2/3 > 2$.

It is straightforward to see that the result of Theorem 1 holds also when the lifting is performed and valid inequality (21) is considered instead of valid inequality (10). As a consequence of Corollary 1, if the number of singletons is 0, then (21) is facet-inducing for the lifted polyhedra. As an example, let us come back to the situation sketched in Figure 1: $C^h_{t4} = \{5, 6\}$ identifies a feasible subpath of length 4. We now assume that $A(5) = \{7\}$ and $A(6) = \emptyset$, i.e. there exist an activity arc, namely arc 7, which covers the same layers as activity arc 5. Let us then y_3 be the variable corresponding to arc 7.

It is straightforward to see that the following vectors

$$\begin{array}{c} (x_1) \\ (x_2) \\ (x_3) \\ (x_4) \\ (y_1) \\ (y_2) \\ (y_3) \end{array} \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

are linearly independent and satisfy constraint $y_1 + y_2 + y_3 \leq 1$ as equality: in fact the multiplier associated with the last column of the matrix is zero (see the last row) and the linear independence of the other 6 vectors has been shown before for the case when $A(5) = \emptyset$.

It is easy to see that the same considerations hold for the case of singletons being present; likewise it is not restrictive to assume that $A(e)$ contains at most one element for any e in $C^h_{t,p+1}$.

5 Computational results

We now report on some computational results obtained by introducing a subset of constraints (10). In fact, it is worth noting that the valid inequalities proposed in the previous section can be arranged into two groups: all those constraints which can be treated implicitly in the preprocessing phase, while building the graph G, belong to first group, whereas the second group, consists of those constraints which have to be explicitly added to ACR. In the computational experiments carried out so far, only the valid inequalities of the first group have been taken into account as explained in the following. Let e be the arc corresponding to a service activity ending on day t

and k be the arc corresponding to a service activity beginning on day $t+1$ which is compatible with e for an employee h. We further assume that the sum of the length of the two activities is greater than p. In such a situation the constraint $x_e^h + x_k^h \leq 1$ holds and we do not build the compatibility arc between arcs e and k. The constraint has thus been considered implicitly in the construction of the graph G. On the contrary it is impossible to consider implicitly the valid inequalities involving more than two variables. Should not the preprocessing phase alone be powerful enough, a bigger set of valid inequalities could be inserted explicitly. Since the number of valid inequalities (10) can be huge, some criteria to dynamically select a good pool of valid inequalities to add, should be identified.

Eliminating the compatibility arc rather than introducing explicitly the valid inequality (10) has a twofold advantage: (i) the number of rows in the linear programming (LP) problem does not increase; (ii) allows to cut off fractional solution which satisfy the valid inequality. For example, for the case of Figure 1, eliminating the arc between the two consecutive activity arcs 5 and 6 allows to cut off the following feasible solution $x_1^h = x_2^h = 0$, $x_3^h = x_4^h = x_5^h = x_6^h = 1/2$, which is not cut off by constraint $x_5^h + x_6^h \leq 1$.

We tested our approach on 7 real instances from the medium size Italian airline which are described in Table 1.

ProbName	**Employees**	**Pairings**	**Reserves**	**Preassigned**	**dMax**	z_{UB}^n	z_{UB}^l
Jul0	24	189	0	0	4	189	503
Oct0	52	244	0	507	4	244	788
Oct1	52	259	0	565	6	259	846
Oct2	52	259	31	565	6	290	908
Oct3	54	259	0	860	6	259	846
Sep0	49	275	0	751	4	275	837
Sep1	50	277	30	760	4	307	904

Table 1: The data set

The problem name, reported in the first column, contains information relative to the month during which the work assignment is to be done. Then for each problem, the number of employees involved, the number of pairings and reserve blocks which should be covered, the number of preassigned activities to be guaranteed, the maximum length of a pairing (expressed in working days) are given. In the last two columns two trivial upper bounds on the number of activities which can be covered and on their total length respectively are shown: in particular z^n_{UB} is given by the sum of the number of pairings and reserves, while z^l_{UB} is obtained as the sum of the length of pairings and reserves which should be covered in the programming period; we assume in fact, that the constant term due to the preassigned activities, which are always guaranteed through constraints (7) in our approach, is not taken into account when evaluating the objective function or its upper bounds. Observe that Jul0 is the smallest instance in the test bed, involving only 24 employees, it is an unstructured problem: no activity is preassigned to the employees thus resulting in a potentially difficult problem to our approach.

In Tables 2 and 3 some results obtained with the commercial linear programming solver CPLEX 7.0 [8] are given. All tests were carried out on a Athlon at 1200Mhz with 512Mb of main memory. For each problem of the test bed, whose name is reported in column 1, we give some data concerning the LP relaxations of ACR ($\overline{ACR}$) and of the problem obtained after the preprocessing phase ($\overline{ACR+PP}$) for each objective function chosen, either the max cardinality case or the max length case: in particular the objective function value, the number of rows, columns and nonzero elements in the LP formulation are shown.

Observe that according to the upper bound given by the LP relaxation, all of the activities can be covered but for the last problem in the test bed (problem Sep1).

An estimate of the number of valid inequalities introduced is given by the difference between the number of columns in the LPs before and after the preprocessing phase respectively: in fact, as we observed above, every time a valid inequality involving two variables is taken into account, a compatibility arc is removed.

The cuts introduced in the preprocessing phase allow to reduce significantly the

	ACR						*ACR + PP*					
ProbName	**Obj**	**Rows**	**Columns**	**Coeff**	**Frac**	**Time**	**Obj**	**Rows**	**Columns**	**Coeff**	**Frac**	**Time**
Jul0	189	11229	39552	107544	279	77.6	189	11157	31368	91176	347	90.05
Oct0	244	18993	63825	179068	66	55.62	244	17615	39816	130537	36	132.06
Oct1	259	17486	57563	162625	87	53.95	259	16142	36183	118712	40	129.59
Oct2	290	19941	72692	195685	227	120.54	290	19483	52003	153141	92	83.43
Oct3	259	13088	40502	117088	379	25.8	259	12179	26739	88685	44	61.88
Sep0	275	12878	43367	122879	333	111.36	275	12183	28248	92258	308	96.82
Sep1	306	15026	55542	150439	499	198.88	306	14780	40352	119641	384	269.28

Table 2: The LP results: the max cardinality case

	ACR						*ACR + PP*					
ProbName	**Obj**	**Rows**	**Columns**	**Coeff**	**Frac**	**Time**	**Obj**	**Rows**	**Columns**	**Coeff**	**Frac**	**Time**
Jul0	503	11229	39552	107544	232	53.93	503	11157	31368	91176	205	63.63
Oct0	788	18993	63825	179068	208	192.47	788	17615	39816	130537	12	113.05
Oct1	846	17486	57563	162625	122	44.83	846	16142	36183	118712	28	45.13
Oct2	908	19941	72692	195685	97	71.99	908	19483	52003	153141	0	178.45
Oct3	846	13088	40502	117088	398	26.15	846	12179	26739	88685	114	19.38
Sep0	837	12878	43367	122879	479	80.25	837	12183	28248	92258	319	65.26
Sep1	901	15026	55542	150439	434	222.92	901	14780	40352	119641	297	214.77

Table 3: The LP results: the max length case

number of columns involved even if such a reduction not always results in a reduction of the computational time required to solve the LP (see columns Time).

Hereafter computational time reported is given in CPU seconds unless otherwise stated.

Furthermore the number of fractional variables in the LP solution, with and without the cuts, is reported (see columns Frac in Tables 2 and 3): the solutions obtained from the LP by adding the cuts are better than the ones obtained without them from the integrality point of view, but for the first problem in the test bed when the aim is to maximize the number of covered activities. The cuts allow to reduce the number of fractional variables in the LP solution, no matter of which objective function is used, but when maximizing the total length a greater reduction seems to be obtained than for the max cardinality case. There are several cases (see for example problems Oct2 and Oct3 in Table 2 and Oct0 and Oct3 in Table 3) where a remarkable improvement in the quality of the LP solution is observed (number of fractional variables reduced by a factor of 20 for problem Oct0). Moreover in one case (problem Oct2 in Table 3) the optimal integer solution is obtained by solving the LP formulation.

Tables 4 and 5 report some performance measures (relative error, number of nodes of the branch and bound tree, time) of CPLEX 7.0 when used to solve the problems to optimality, with and without the preprocessing phase. A maximum time limit of 15 CPU hours has been fixed for the most difficult problems in the test bed and the best solution found when this limit is exceeded is reported. The relative error ε_r is computed as $\frac{z(LP)-z(IP)}{z(IP)}$.

	ACR			$ACR+PP$		
ProbName	$\%\varepsilon_r$	**Nodes**	**Time**	$\%\varepsilon_r$	**Nodes**	**Time**
Jul0	4.42	1300	$15h$	0	29	$33'$
Oct0	0	10+	354.74	0	0+	116.18
Oct1	0	17	326.05	0	5	204.52
Oct2	0	3	248.67	0	0	155.9
Oct3	0	69	666.56	0	0	48.75
Sep0	3.38	13300	$15h$	0	2941	$8h$
Sep1	2.04	2200	$15h$	0	878	$14h$

Table 4: The IP results: the max cardinality case

	ACR			$ACR+PP$		
ProbName	$\%\varepsilon_r$	**Nodes**	**Time**	$\%\varepsilon_r$	**Nodes**	**Time**
Jul0	3.29	700	$15h$	0	143	$1h\ 40'$
Oct0	0	19	464.41	0	0+	134.26
Oct1	0	19	270.08	0	0+	113.03
Oct2	0	0	169.08	0	0	212.42
Oct3	0	47	617.83	0	0+	54.1
Sep0	1.7	13300	$15h$	0	239	$1h\ 46'$
Sep1	1.29	3100	$15h$	0.11	240+	$4h\ 15'$

Table 5: The IP results: the max length case

As Table 4 and 5 clearly show most of the instances can be solved to optimality within reasonable computational time and after that very few nodes of the enumeration tree have been explored (the symbol + near the number of nodes is used to indicate that the optimal solution has been found via the node heuristic procedure of CPLEX); moreover when the preprocessing phase is done, several instances are solved at the root node. Note that for all the test problems but the last, the gap between the upper bound given by the LP relaxation and the optimal integer solution is zero:

such a fact seems to indicate that the model proposed fits well the rostering problem studied.

The validity of the approach proposed however, comes into evidence more clearly as the difficulty of the problem grows: as far as the max cardinality case is concerned, by introducing the cuts in the preprocessing phase, the unstructured problem Jul0 is solved to optimality within half an hour while without the cuts, CPLEX exceeds the time limit without providing an optimal solution and after 2 hours a quite poor feasible solution (relative error over 4%) only is reported. Also for problems Sep0 and Sep1 (see Table 4) CPLEX fails to provide an optimal solution while with the cuts an optimal solution is reported within 8 and 14 hours respectively. Anyway we would stress here that quite good solutions (relative error not over 0.7%) are always reported within an hour and a half in the max cardinality case when the preprocessing phase is enabled (see Table 6).

Even more promising results seem to be reported by the max length case: for the most difficult problems in the test bed (Sep0 and Sep1) CPLEX without the cuts, still fails to provide the optimal solution within 15 hours, even if better solutions, in terms of relative error, than the ones obtained in the max cardinality case, are given. On the other hand the time required to solve the problem to optimality when the preprocessing phase is enabled, is remarkably less than the one required in the max cardinality case: less than 2 hours for problem Sep0 against the 8 hours required by the max cardinality case, and about 4 hours for Sep1 against 14 hours. Only for the first problem in the test bed, the performance has been worse: the time required by the max length case has grown in fact by a factor 3 over the time required by the max cardinality case; nevertheless the computational results are still good compared with the ones obtained without the preprocessing phase.

Finally the computational experiments seem also to indicate that the max cardinality case outperforms the max length case in terms of quality of the first feasible solution obtained. We have run a truncated version of the Branch and Bound (B&B) in which

CPLEX is stopped as soon as an integer solution is found. Computational results are shown in Table 6 and 7 for the max cardinality case and the max length case respectively. When the preprocessing phase is disabled, the max length case provide more accurate solutions compared with the ones obtained by the max cardinality case, even if the time required in order to compute them, can increase significantly (see for example Jul0). On the other hand, when the preprocessing phase is enabled, the max cardinality case outperforms the max length case both in terms of quality of the solution found and of computational time: for the most difficult problems in the test bed the time required to get a feasible solution in the former case is greater than in the latter case. Moreover only in one case (Sep0) the max cardinality case has produced a solution with a relative error bigger than the one obtained for the max length case, requiring however half an hour against an hour.

	ACR			$ACR + PP$		
ProbName	$\%\varepsilon_r$	**Nodes**	**Time**	$\%\varepsilon_r$	**Nodes**	**Time**
Jul0	4.42	178	$2h$	0	29	$33'$
Oct0	0	10+	354.74	0	0+	116.18
Oct1	0	17	326.05	0	5	204.52
Oct2	0	3	248.67	0	0	155.9
Oct3	0	69	666.56	0	0	48.75
Sep0	3.38	436	$1h\ 30'$	0.73	48	$30'$
Sep1	2.08	397	$4h\ 20'$	0.33	64	$1h\ 25'$

Table 6: The truncated B&B results: the max cardinality case

	ACR			$ACR + PP$		
ProbName	$\%\varepsilon_r$	**Nodes**	**Time**	$\%\varepsilon_r$	**Nodes**	**Time**
Jul0	3.29	236	$5h\ 10'$	0.6	68	$1h$
Oct0	0	19	464.41	0	0+	134.26
Oct1	0	19	270.08	0	0+	113.03
Oct2	0	0	169.08	0	0	212.42
Oct3	0	47	617.83	0	0+	54.1
Sep0	1.7	786	$1h\ 40'$	0.36	79	$1h$
Sep1	1.35	273	$3h$	1.35	93	$1h\ 40'$

Table 7: The truncated B&B results: the max length case

6 Conclusion

In this work a 0-1 Multicommodity Flow model with additional constraints has been proposed for the crew rostering problem in an airline. Some issues concerning the polyhedral structure of the model proposed have been investigated. Some families of valid inequalities have been proposed and their properties discussed. Particular attention has been given to the structure of polyhedra resulting from the decomposition of the original polyhedron, for which the valid inequalities proposed are shown to be facet-inducing. A preprocessing phase has been performed which consists of taking into account only those valid inequalities which can be treated implicitly in the construction of the compatibility graph. Computational results carried out with CPLEX seem to show the validity of the approach proposed, at least on the set of real instances solved. Good results have been obtained for medium scale problems by enabling the preprocessing phase. As an example, the optimal solution for an instance with 40352 columns, 14780 rows and 119641 nonzero elements has been obtained in about 4 CPU hours, while, without the cuts, CPLEX fails to provide the optimal solution within 15 hours. Anyway quite good solutions (relative error not over 0.7%) have been reported within an hour and an half, when the preprocessing phase is done, for all the real instances solved. We would like to stress the fact that for almost all of the test problems, the gap between the upper bound given by the Linear Programming relaxation and the optimal integer solution is zero: generally speaking such a fact indicates either that the model proposed fits well the problem studied or the problems are not too difficult to solve. However, in our case, the results obtained with CPLEX without the preprocessing phase, seem to indicate that some of the instances are really hard to solve. We would like to point out that the effectiveness of the preprocessing phase depends also on the length of the service activities with respect to the maximum number of working days between two rest activities. Let us compare medium and long haul versus short haul: in the former case the probability of having covers composed only by two service activities is higher than in the latter case and the preprocessing phase takes advantage of this structure, resulting in a remarkable reduction of the computational time required to get the optimal solution. On the other hand, in short haul covers are generally composed by more than two activities and the number of valid inequalities (10) which should be generated could be huge. To this end a cutting plane approach in which the valid inequalities are dynamically generated seems to be more suitable than an approach where the covers are computed at the very beginning and inserted in the LP formulation from scratch. In such cases, in order to avoid the insertion of the cuts be too burdensome, clever rules should be devised which dynamically select a good pool of valid inequalities to add. Moreover also on large scale problems a bigger subset of valid inequalities than the one currently put into the model, could be used in order to get hopefully quite good solutions within reasonable computational time. These are future research lines we would like to pursue.

References

[1] J. Byrne. A Preferential Bidding System for Technical Aircrew. *AGIFORS Symposium Proceeding*, 28:87–99, 1988.

[2] P. Carraresi and G. Gallo. A Multi-Level Bottleneck Assignment Approach to the Bus Drivers' Rostering Problem. *European Journal of Operational Research*, 16:163–173, 1984.

[3] M. Gamache and F. Soumis. A Method for Optimally Solving the Rostering Problem. In G. Yu, editor, *Operations Research in the Airline Industry*, pages 124–157. Kluwer, Boston, 1998.

[4] M. Gamache, F. Soumis, G. Marquis, and J. Desrosiers. A Column Generation Approach for Large-Scale Aircrew Rostering Problems. *Operations Research*, 47(2):247–263, 1999.

[5] M. Gamache, F. Soumis, D. Villeneuve, J. Desrosiers, and É. Gélinas. The Preferential Bidding System at Air Canada. *Transportation Science*, 32(3):246–255, 1998.

[6] C. Giafferri, J.P. Hamon, and J.G. Lengline. Automatic Monthly Assignment of Medium-Haul Cabin Crew. *AGIFORS Symposium Proceeding*, 22:69–95, 1982.

[7] W. Glanert. A Timetable Approach to the Assignment of Pilots to Rotations. *AGIFORS Symposium Proceeding*, 24:369–391, 1984.

[8] ILOG. *ILOG Cplex 7.0, User's Manual*, 2000.

[9] N. Kohl and E. Karisch. Airline Crew Rostering: Problem Types, Modeling, and Optimization. Technical Report CRTR-2001-1, Carmen Systems, 2001. Available at `http://www.carmen.se/research_development/-research_reports.htm`.

[10] F. Marchettini. Automatic Monthly Cabin Crew Rostering Procedure. *AGIFORS Symposium Proceeding*, 20:23–59, 1980.

[11] R. Moore, J. Evans, and H. Ngo. Computerized Tailored Blocking. *AGIFORS Symposium Proceeding*, 18:343–361, 1978.

[12] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization.* John Wiley & Sons, 1988.

[13] D.M. Ryan. The Solution of Massive Generalized Set Partitioning Problems in Air Crew Rostering. *Journal of the Operational Research Society*, 43:459–467, 1992.

[14] D. Sarra. The Automatic Assignment Model. *AGIFORS Symposium Proceeding*, 28:23–37, 1988.