OPERATIONS RESEARCH
Vol. 52, No. 4, July–August 2004, pp. 583–596
ISSN 0030-364X | EISSN 1526-5463 | 04 | 5204 | 0583

informs

DOI 10.1287/opre.1040.0110

A Multicommodity Flow Approach to the Crew Rostering Problem

Paola Cappanera, Giorgio Gallo
Dipartimento di Informatica, Università di Pisa, Via F. Buonarroti 2, 56127 Pisa, Italy
{cappaner@di.unipi.it, gallo@di.unipi.it}

The problem of finding a work assignment for airline crew members in a given time horizon is addressed. In the literature this problem is usually referred to as the *airline crew rostering problem*. It consists of constructing monthly schedules for crew members by assigning them pairings, rest periods, annual and sick leave, 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 still satisfying union conventions. A preprocessing phase is performed that reduces the dimension of the graph. To tighten the linear programming formulation of our model, we propose some families of valid inequalities that have proved to be computationally effective. Some of them can be treated implicitly when constructing the graph. Computational results obtained with a commercial integer programming solver (CPLEX) are analyzed.

Subject classifications: transportation, scheduling, personnel: crew rostering; programming: integer; cutting plane/facet: valid inequalities.
Area of review: Transportation.
History: Received October 2001; revisions received January 2003, April 2003; accepted May 2003.

1. Introduction

The problem of finding a work assignment for crew members in a given time period is addressed. When dealing with 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. A pairing is a sequence of work assignments, possibly including embedded breaks, that starts and ends at a given crew base. Second, monthly schedules for crew members are constructed by assigning to each crew a set of 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 leave, 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.

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 in most North American companies. The rostering method (Carraresi and Gallo 1984, Ryan 1992, Gamache et al. 1999) consists of 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 (Moore et al. 1978, Byrne 1988, Gamache et al. 1998), schedules are constructed that 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 also be found in crew rostering problems at railway and extraurban transportation companies (see Kohl and Karisch 2004), to the extent that our approach may also be suitable in these settings.

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 a 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 (Marchettini 1980, Glanert 1984), the

day-by-day assignment of pairings to employees selected from a pool of available crew members (Sarra 1988), and the sequential construction of rosters, employee by employee (Moore et al. 1978, Byrne 1988). Combinations of these approaches have also been developed (Giafferri et al. 1982). Each of the above methods has several disadvantages, the main one being that they all lack 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 that take more advantage of the structure of the problem have been proposed and are based on a generalized set-partitioning model. To solve the rostering problem, two main methods have been presented. The first consists of generating a priori a set of feasible rosters for each employee and solving the resulting problem by integer programming techniques (Ryan 1992). The second is a column generation approach (Gamache and Soumis 1998). The performance of the first method very much depends on the goodness of the rosters selected, while a standard column generation approach is usually not able to tackle large-scale problems. However, in Gamache et al. (1999), interesting algorithmic control strategies have been proposed and tested to accelerate the solution time of a column generation approach.

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 on its structure by exploiting some of its peculiarities. Here the airline crew rostering problem, hereafter 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 still satisfying 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 that 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, and so forth. These are typical rules and regulations that do not depend on the particular case study (see Kohl and Karisch 2004 for a comprehensive description of real world airline crew rostering problems): for example, 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 noncovered activities; however, the objective function generally varies from one airline to another and often includes terms that deal with the even distribution of the workload among the crew members. A workload can be expressed in terms of the number of monthly rest periods, the number of reserve blocks, flight hours, service hours, and other quantitative indicators. However, over the years, equidistribution concerns have been inserted directly into the collective agreements and are thus treated as constraints. This is why only the minimization of noncovered activities is here taken into account in the objective function.

This paper is organized as follows. In §2, a detailed description of the rostering problem is presented, and a 0–1 multicommodity flow model with additional constraints is proposed. In §3, some families of valid inequalities are introduced and their properties discussed to tighten the linear programming (LP) formulation of the model provided. In §4, the valid inequalities proposed are strengthened. Finally, in §5, computational results are reviewed and some conclusions are drawn in §6.

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 that 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, a set of n rosters has to be found, one for each employee. In our case the rostering problem deals with three kinds of crew members belonging to the same base: 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 one 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 on day t. Hereafter, τ_i will denote 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, an origin node o^h and a destination node d^h are also 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 made up of 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 example, a union activity, a training period, annual leave, or 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 exactly 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 on 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 days 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 $t+1$; in 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 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 $i \in N$ and for each employee h:

$F_s(i,h)$ the *forward star* of i, i.e., the set of arcs outgoing from i which can be used by commodity h, and
$B_s(i,h)$ the *backward star* of i, i.e., the set of arcs entering into i which can be used by commodity h.

More formally, we have

- if i is the beginning of an activity $e \in A$ compatible with h, then $F_s(i,h) = \{e\}$;
- if i is the ending of an activity $e \in A$, then $F_s(i,h) = \{(i,j) \in C \text{ s.t. } j \text{ is the beginning of an activity compatible with } e \text{ and } h\}$.

$B_s(i,h)$ 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 example, in our case, a union clause stipulates that there can 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 graph G, described above, 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 achieved by connecting, for each employee h, node o^h to the node that corresponds to the initial time of the first relevant activity assigned to 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 because 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 only employed recently), then o^h is connected to all the activities that can be performed by h on the first day of the current programming period.

Moreover, employee h can be assigned an activity that crosses 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, all the nodes in G 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 to formulate the union clauses and collective agreement 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 (in our case equal to one for pairings, reserve blocks, and preassigned activities),
Q_e the set of qualifications required by e, and
F_e the set of activities which cannot be performed immediately after e (forbidden pairings).

There are a number of restrictions that cannot be considered implicitly in the construction of graph G because 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^h is given, which depends on employee h as well. There are restrictions on the maximum *flight* and *service times* an employee can accumulate in each calendar week of the programming period and in the entire month, and restrictions on the minimum number of monthly rest periods (nine in our case study).

The *flight time* of an activity is the time spent by an employee onboard an 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 company base. 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 training periods.

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

In addition, for each employee h the following attributes are known:

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

Let us now analyse in depth the way the compatibility arc set is constructed. Usually, the linking or compatibility arcs depend on the employee h to whom activities are to be assigned. In any case, 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 and h, $F_s(i, h)$, can be constructed. $F_s(i, h)$ is made up of all arcs (i, j), where j is the beginning node of an activity e', such that the following conditions are satisfied: (i) $\tau_j + 24 \geqslant \tau_i + \tau_r$; (ii) $e' \notin F_e$; (iii) $Q_{e'} \subseteq Q^h$; and (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* that 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 that is compatible with e (this happens, for example, when e is a pairing terminating very late at night), activity e is linked to the *physiologic rest duty*. When this happens, $F_s(i, h)$ contains only one 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. To avoid misunderstandings, note that physiologic rest period and physiologic rest duty have two different meanings: The first is the minimum number of hours elapsed between two consecutive work days, while the latter is a particular duty in which an employee takes a rest because he/she finished his/her shift very late at night, but it is a work day and it must not be counted as a rest day.

Note that when formulating ACR, all the above constraints can be treated implicitly in the construction of graph G. Both the rest day and the physiologic rest duty are activities that 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 leave and training periods. Observe that the activities covered by an employee in the last p days of the previous programming period must also be considered as preassigned activities. Thus, let

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

Let us recall the following parameters:

p the maximum number of working days between two rest activities, and
$d_{\max}$ the maximum length of an activity.

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

Let us then define

S_t as the set of arcs corresponding to rest activities which cover layers $t, t+1, \ldots, t+p$, and
t^h as the day following the last rest day assigned to h in the previous programming period.

ACR can be formulated as

$$\text{(ACR)} \quad \min \sum_{e \in E} \gamma_e \sum_h x_e^h$$

$$\text{s.t.} \sum_{e \in F_s(o^h, h)} x_e^h = 1 \quad \forall h, \tag{1}$$

$$\sum_{e \in B_s(d^h, h)} x_e^h = 1 \quad \forall h, \tag{2}$$

$$\sum_{e \in B_s(i, h)} x_e^h - \sum_{e \in F_s(i, h)} x_e^h = 0 \qquad \forall h \ \forall i \in N, \tag{3}$$

$$\sum_h x_e^h \leqslant c_e \quad \forall e \in A, \tag{4}$$

$$\sum_{e \in S_t} x_e^h \geqslant 1 \qquad \forall h \ \forall t \in \{t^h, t^h + 1, \ldots, m\}, \tag{5}$$

$$\sum_{e \in A} r_{ek} x_e^h \leqslant r_k^h \quad \forall h \ \forall k, \tag{6}$$

$$x_e^h = 1 \quad \forall h \ \forall e \in P^h, \tag{7}$$

$$x_e^h = 0 \quad \forall h \ \forall e \in F^h, \tag{8}$$

$$x_e^h \in \{0, 1\} \quad \forall h \ \forall e \in E, \tag{9}$$

where the costs γ_e are defined either as

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

or

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

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 that will be assigned to extra staff 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 constraint 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. Constraint (4) limits the total quantity of flow on activity arc $e \in A$, regardless of the commodity. It ensures 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 ensure 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.

Constraint (5) thus guarantees 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. $t \leqslant m$ because 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, because graph G has been extended to the left. Moreover, observe that, for each employee h, the actual number of constraint (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 thus guaranteed.

Here only daily rest periods are assumed. Longer leave periods are represented as sequences of daily periods; as an example a 15-day leave is represented by 15 consecutive rest periods of length equal to one.

Resource Constraints. Constraint (6) is the resource constraint that, for each employee h, ensures that the level of consumption of each resource k along a path must not exceed the threshold r_k^h. Examples of resources are weekly and monthly flights and service times, the number of rest periods, reserve blocks, and so forth.

Preassigned Activities Constraints. For each employee h constraint (7) fixes to one the flow along all the arcs corresponding to preassigned activities.

Forbidden Assignments Constraints. For each employee h constraint (8) fixes to zero the flow on all the arcs corresponding to activities that 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 $(1), \ldots, (8)$ and by the continuous relaxation of constraint (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 constraint (4), which link together the otherwise separable single commodity flows. If constraint (4) was not present, ACR could be broken down into n subproblems, each of them defined by all the constraints relative to a given h. But looking at the actual structure of polyhedron (relative to commodity) h, we observe that it can also 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 constraint (6), and
- P_3 the polyhedron given by the working day constraint (5), which can be arranged so as to exhibit a (block) diagonal structure with a bandwidth equal to $p+1$.

To tighten the LP formulation of ACR, it is worth studying the structure of the convex hull of the 0–1 points in $P_1 \cap P_2 \cap P_3$. In particular, we will 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 §5 show. In this section, some families of valid inequalities for ACR are presented, which are subsequently 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. For any h, t, and d, set $c \subseteq A^h$ is called a cover for employee h if it identifies a subpath on G^h, i.e., a sequence of nonoverlapping activities, pairwise connected by compatibility arcs, which starts at day t and covers d consecutive working days. C^h_{td} denotes the collection of all such covers.

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

We now focus on constraint (5), which ensures that for each employee h the number of consecutive working days between two rest periods is at most p. For any $c \in C^h_{t,\,p+1}$, the inequality

$$\sum_{e \in c} x^h_e \leqslant |c| - 1 \tag{10}$$

is valid for ACR because no feasible solution exists with $x^h_e = 1 \ \forall e \in c$: In fact, should such a solution exist, employee h would have to perform $p+1$ consecutive working days without a rest, thus violating constraint (5).

We now show an example to better understand the notation used: suppose that $p = 3$ and consider a time window of four 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 four rest activities, namely arcs $1, 2, 3, 4$, one for each day, and two service activities, both of them two days long, namely arcs 5 and 6. Such a situation is shown in Figure 1, where dashed lines represent the compatibility arcs. Because there exists a compatibility arc between activities 5 and 6, $c = \{5, 6\}$, with $c \in C^h_{t4}$, identifies a feasible subpath of length 4, which violates the working day constraint. In a situation like this,

$$x^h_5 + x^h_6 \leqslant 1$$

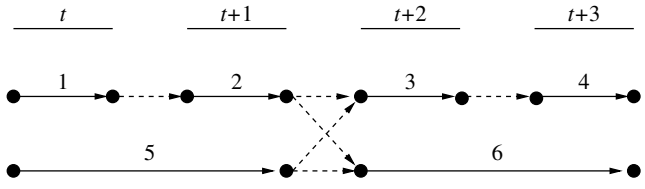
is a valid inequality for ACR, and guarantees that activities 5 and 6 cannot both be assigned to employee h.

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

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

In fact, because they both belong to c, they correspond to services that can be assigned to the same roster and therefore do not overlap.

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



Clearly, for any $c \in C^h_{t,\,p+1}$, $\{R(e) : e \in c\}$ defines a partition of the arcs in S_t. Then, the following constraints hold:

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

$$x^h_e + x^h_k \leqslant 1 \quad \forall k \in R(e)\ \forall e \in c, \tag{11}$$

where constraint (11) is implied by the flow conservation constraints relative to commodity h.

For any $c \in C^h_{t,\,p+1}$, let us define $P^h_t(c)$ as

$$P^h_t(c) = \left\{x^h_e \text{ s.t. } (5), (11),\ x^h_e \in \{0, 1\}\ \forall e \in c \cup S_t\right\}.$$

For example, going 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 corresponding valid inequality (10) cuts off the following fractional solution:

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

which is feasible to the linear relaxation of $P^h_t(c)$.

Let us then denote by $\bar{P}^h_t(c)$ the continuous relaxation of $P^h_t(c)$, by $\dim(\bar{P}^h_t(c))$ the dimension of polyhedron $\bar{P}^h_t(c)$, and finally, by $\text{conv}(P^h_t(c))$ the convex hull of $P^h_t(c)$.

PROPOSITION 1. *For any $c \in C^h_{t,\,p+1}$, $\bar{P}^h_t(c)$ is full dimensional.*

PROOF. Let us assume without loss of generality (w.l.g.) that $|S_t| = p+1$. It is straightforward to see that point $\bar{x}^h$ is defined as follows:

$$\bar{x}^h_e = \begin{cases} 1-\varepsilon & \text{if } e \in S_t, \\ \varepsilon/2 & \text{if } e \in c \end{cases}$$

is an interior point of $\bar{P}^h_t(c)$ for each ε such that $0 < \varepsilon < p/(p+1)$. In fact, the box constraints $0 \leqslant x^h_e \leqslant 1$ are never satisfied as equality by $\bar{x}^h_e$ because $\varepsilon > 0$; moreover, $\bar{x}^h_e$ satisfies constraints (5) and (11) as strict inequality, respectively, for $\varepsilon < p/(p+1)$ and $\varepsilon > 0$. $\bar{P}^h_t(c)$ is thus full dimensional and $\dim(\bar{P}^h_t(c)) = p + 1 + |c|$. □

Now we will examine in more detail the valid inequalities shown so far. In particular, we address the question whether such inequalities are facet inducing for $\text{conv}(P^h_t(c))$ or not.

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

LEMMA 1. *Given a cover $c \in C^h_{t,\,p+1}$, for each nonsingleton activity e in c, the maximum number of linearly independent vectors of $P^h_t(c)$ which satisfies (10) as equality by fixing the corresponding variable x^h_e to zero is equal to $l_e + 1$.*

PROOF. First, we show one possible way to obtain a set of $l_e + 1$ vectors, then their linear independence is proven.

Given service activity e in c, $R(e)$, which is the set of rest arcs covering the same layers as e, has cardinality l_e, and r_k denotes the kth 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 $s \in c$, 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 $c\backslash\{e\}$ have value one, while all the other components have value zero. The rationale is that vectors v_k^e are obtained by fixing the variable relative to service activity e to zero. Then, all variables relative to all the service activities except activity e are set to one to satisfy (10) as equality. Constraint (11) ensures that $x_r^h = 0\ \forall s \in c\backslash\{e\}$, $\forall r \in R(s)$. Thus, to satisfy constraint (5), we fix to one, only one of the rest variables in $R(e)$: there are l_e ways of choosing such a variable and thus l_e vectors v_k^e are obtained. In particular, in vector v_k^e the rest variable relative to the kth element of $R(e)$ is fixed to one.

Finally, $v_{l_e+1}^e$ is defined as follows. All the components corresponding to elements of $R(e)$ and of $c\backslash\{e\}$ are fixed to one, while all the other components have value zero. Without loss of generality, we can assume that the first $p+1$ components of v_k^e refer to the rest arcs and the last $|c|$ ones correspond to the service activities belonging to cover c. Then, the $l_e + 1$ vectors can be arranged to form the following matrix:

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

where w.l.g. we assume that activity e is the first activity in the cover. Observe that the first two blocks relate to the rest variables; the former comprises the l_e rest variables in $R(e)$, while the latter relates to all the other rest variables. The third block consists of a single row relating to service activity e, which is set to 0; finally, the last block relates to the other service activities, besides e, which are set to 1. We observe that $\text{rank}(M_e) \leqslant l_e + 1$, but it is easy to show that $\text{rank}(M_e)$ is equal to $l_e + 1$. In fact, the submatrix of M_e obtained by picking the first l_e rows of M_e and an all-ones row from the last block, i.e.,

$$\begin{bmatrix} I_{l_e} & 1 \\ 1 & 1 \end{bmatrix}$$

by a proper linear combination of the rows, results in the following diagonal matrix with nonzero diagonal elements:

$$\begin{bmatrix} I_{l_e} & 1 \\ 0 & l_e - 1 \end{bmatrix}.$$

Finally, we observe that any other feasible vector to $P_t^h(c)$ which differs from the columns in M_e only in terms of the first l_e components is linearly dependent on them, and this completes the proof. □

Because linear independence implies affine independence, while the converse is not true, the $l_e + 1$ vectors provided in the proof of Lemma 1 are affinely independent. This fact does not exclude that a number of affinely independent vectors greater than $l_e + 1$ exists. However, it is possible to show that the maximum number of affinely independent vectors which satisfies the hypothesis of Lemma 1 is indeed $l_e + 1$. For the sake of convenience, we recall the definition of affine independence.

Definition 2. A set of points $v^1, \ldots, v^k \in R^n$ is affinely independent if the unique solution of $\sum_{i=1}^k \alpha_i v^i = 0$, $\sum_{i=1}^k \alpha_i = 0$ is $\alpha_i = 0$ for $i \in \{1, \ldots, k\}$.

Then, the following result holds.

Lemma 2. *Given a cover $c \in C_{t,\,p+1}^h$, for each nonsingleton activity e in c, the maximum number of affinely independent vectors of $P_t^h(c)$, which satisfies* (10) *as equality by fixing the corresponding variable x_e^h to zero, is equal to $l_e + 1$.*

Proof. We assume by contradiction that in addition to the $l_e + 1$ vectors provided in Lemma 1, a vector $v_{l_e+2}^e$ exists that is feasible for $P_t^h(c)$ and satisfies (10) as equality with x_e^h fixed to zero such that $v_1^e, \ldots, v_{l_e+2}^e$ are affinely independent. From the definition of affine independence, it follows that the only solution of system

$$\sum_{i=1}^{l_e+2} \alpha_i v_i^e = 0, \qquad \sum_{i=1}^{l_e+2} \alpha_i = 0 \tag{12}$$

is $\alpha_i = 0$ for all $i \in \{1, \ldots, l_e + 2\}$. From Lemma 1 we have that the maximum number of linearly independent vectors of $P_t^h(c)$, which satisfies (10) as equality when x_e^h is fixed to zero, is $l_e + 1$; then

$$\exists \lambda \neq 0 \quad \text{s.t.} \quad \sum_{i=1}^{l_e+2} \lambda_i v_i^e = 0.$$

By construction, the last component of vectors v_i^e for $i \in \{1, \ldots, l_e + 2\}$ is one, so $\sum_i \lambda_i v_i^e = 0$ implies $\sum_i \lambda_i = 0$. Hence, there exists a nonnull solution λ of system (12) and this contradicts the assumption. □

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

Proof. As shown in Proposition 1, $\bar{P}_t^h(c)$ is full dimensional; to prove that (10) is a face of $\text{conv}(P_t^h(c))$ of

dimension $\dim(\bar{P}_t^h(c)) - k - 1$, we provide $\dim(\bar{P}_t^h(c)) - k$ linearly independent vectors, that are feasible to $P_t^h(c)$ and satisfy (10) as equality. Valid equality (10) is satisfied as equality, if in turn one of the $|c|$ service activities in c is set to zero while all the other activities in the cover are set to one. Lemma 1 proves that, for a given nonsingleton e in c, we can construct $l_e + 1$ linearly independent vectors with $x_e^h = 0$.

Let us now examine the case of a singleton e: If x_e^h is set to 0, then x_j^h is set to one $\forall j \in c$ with $j \neq e$ 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 relating 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 nonsingleton e and a vector in correspondence to each singleton. The resulting matrix is shown below, with the matrix obtained by linear combinations of its rows. Let q be the number of nonsingleton activities in the cover, i.e., $q = |c| - k$ and let us assume w.l.g. that the elements in the cover are ordered so that the first q ones correspond to nonsingleton activities. We also use the notation b_{rc} with $b \in \{0, 1\}$ to denote a matrix with r rows and c columns, and the elements are all equal to b.

$$\left[\begin{array}{cc|c} I_{l_1} & 1_{l_1} & \\ & \ddots & 0_{p+1-k\times k} \\ & I_{l_q} \quad 1_{l_q} & \\ \hline \multicolumn{2}{c|}{0_{k\times\dim(\bar{P}_t^h(c))-2k}} & I_k \\ \hline \multicolumn{2}{c|}{B} & 1_{q\times k} \\ \hline \multicolumn{2}{c|}{1_{k\times\dim(\bar{P}_t^h(c))-2k}} & 1_{k\times k} - I_k \end{array}\right]$$

$$\left[\begin{array}{cc|c} I_{l_1} & 1_{l_1} & \\ & \ddots & 0_{p+1-k\times k} \\ & I_{l_q} \quad 1_{l_q} & \\ \hline \multicolumn{2}{c|}{B} & 0_{q\times k} \\ \hline \multicolumn{2}{c|}{1_{k\times\dim(\bar{P}_t^h(c))-2k}} & 0_{k\times k} \\ \hline \multicolumn{2}{c|}{0_{k\times\dim(\bar{P}_t^h(c))-2k}} & I_k \end{array}\right].$$

By definition, in the top matrix, for each singleton e the row (second block) corresponding to the unique rest variable in $R(e)$ contains only a 1 in the column relating to e. By subtracting such k rows from all the last $|c|$ rows in the top matrix and shifting the k rows to the bottom of the matrix, we obtain the bottom matrix.

Let us now analyse submatrix B in detail. This matrix has a row $r(e)$ for each nonsingleton activity e. Readers may recall that this block was constructed by providing for each nonsingleton e, $l_e + 1$ vectors with $x_e^h = 0$. Each row $r(e)$ in B is thus an all-ones row apart from positions corresponding to the $l_e + 1$ vectors. Given a nonsingleton activity e, we can thus substitute row $[r(e), \mathbf{0}]$ with one of the rows in $[1_{k\times\dim(\bar{P}_t^h(c))-2k}, \mathbf{0}]$ minus row $[r(e), \mathbf{0}]$, thus obtaining an 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 immediately after the block $[I_{l_e}, 1_{l_e}]$. By repeating this procedure for each nonsingleton e and exploiting the linear combinations used to prove Lemma 1, we obtain the following matrix:

$$\left.\begin{array}{l} \left[\begin{array}{cc|c} I_{l_1} & 1_{l_1} & \\ 0 & l_1 - 1 & \\ & \ddots & 0_{\dim(\bar{P}_t^h(c))-2k\times k} \\ & I_{l_q} \quad 1_{l_q} & \\ & 0 \quad l_q - 1 & \\ \hline \multicolumn{2}{c|}{0_{k\times\dim(\bar{P}_t^h(c))-2k}} & I_k \\ \hline \multicolumn{2}{c|}{1_{k\times\dim(\bar{P}_t^h(c))-2k}} & 0_{k\times k} \end{array}\right] \end{array}\right\} \dim(\bar{P}_t^h(c)) - k.$$

The submatrix obtained by eliminating the last k rows is a diagonal block matrix with nonzero diagonal elements, and this completes the proof. □

Let us now show an example to better understand how the vectors are generated and how their linear independence can be proven. For the sake of convenience, hereafter when 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:

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \geqslant 1 \tag{13}$$

$$x_1 + y_1 \leqslant 1 \tag{14}$$

$$x_2 + y_1 \leqslant 1 \tag{15}$$

$$x_3 + y_1 \leqslant 1 \tag{16}$$

$$x_4 + y_2 \leqslant 1 \tag{17}$$

$$x_5 + y_2 \leqslant 1 \tag{18}$$

$$x_6 + y_3 \leqslant 1, \tag{19}$$

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

$$P = \big\{(x, y) \text{ s.t. } (13), (14), (15), (16), (17), (18), (19),\ x \in \{0, 1\}^6, y \in \{0, 1\}^3\big\}.$$

Observe that by summing up constraints involving both rest variables x and service variables y, and subtracting inequality (13) to the constraint thus obtained, we get the inequality $3y_1 + 2y_2 + y_3 \leqslant 5$, which is weaker than the valid

inequality $y_1 + y_2 + y_3 \leqslant 2$. Such an inequality cuts off, for example, the following solution feasible to $\bar{P}$:

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

By applying the procedure described in Theorem 1, we obtain the following matrix, where eight vectors are given column by column, as explained before. Beside each row the name of the corresponding variable is shown:

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

For example, by fixing $y_1 = 0$ (and consequently $y_2 = y_3 = 1$) we are able to provide four (because $l_1 = 3$) linearly independent vectors, which are the first four columns of the matrix above. In this example, matrix

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

has two rows corresponding to nonsingleton activities y_1 and y_2. By subtracting the row corresponding to x_6 from the last three rows and moving it to the bottom of the matrix, we obtain the top matrix:

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

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

The bottom matrix was obtained by applying the following linear operations to the top matrix: $y_1 \leftarrow y_3 - y_1$ and $y_2 \leftarrow y_3 - y_2$. We observe that the submatrix obtained from the bottom matrix by eliminating the last row exhibits a diagonal block structure with nonzero diagonal elements; its eight columns are linearly independent, and this shows that $y_1 + y_2 + y_3 \leqslant 2$ is a face of dimension 7.

COROLLARY 1. *For any $c \in C^h_{t,\,p+1}$, if the number of singletons in cover c is equal to* 0*, the valid inequality* (10) *is a facet for* $\operatorname{conv}(P^h_t(c))$.

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

PROOF. By summing up constraint (11), we get the inequality

$$\sum_{e \in c} l_e x^h_e + \sum_{e \in c} \sum_{k \in R(e)} x^h_k \leqslant p + 1,$$

which is the same as

$$\sum_{e \in c} l_e x^h_e + \sum_{e \in S_t} x^h_e \leqslant p + 1,$$

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

$$\sum_{e \in c} l_e x^h_e \leqslant p. \tag{20}$$

Observe that when $l_e = 1\ \forall e \in c$, valid inequality (10) is exactly the same as constraint (20) because in this case $|c| - 1 = p$. □

Indeed, we can strengthen the result above by showing that when the cover is made up only of singletons, constraints defining $\bar{P}^h_t(c)$ fully describe $\operatorname{conv}(P^h_t(c))$.

COROLLARY 3. *For any $c \in C^h_{t,\,p+1}$, if $l_e = 1\ \forall e \in c$, then* $\bar{P}^h_t(c) = \operatorname{conv}(P^h_t(c))$.

It is in fact straightforward to see that the constraint matrix defining $\bar{P}^h_t(c)$ is totally unimodular.

We will now outline an alternative way of getting valid inequality (10) from constraint (20). To do this, we will report some basic results on polyhedra and some standard definitions (for more details see Nemhauser and Wolsey 1988).

Consider a 0–1 knapsack problem, and let

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

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

DEFINITION 3. Given a set $C \subseteq \{1, \ldots, n\}$, we say that C is an independent set if $\sum_{j \in C} a_j \leqslant b$; otherwise C is a dependent set.

PROPOSITION 2. *If C is a dependent set, then*

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

is a valid inequality for F.

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

DEFINITION 4. A dependent set is minimal if all of its subsets are independent.

As far as set $c \in C^h_{t,\,p+1}$ is concerned, it is sufficient to remove an element from it to satisfy constraint (20).

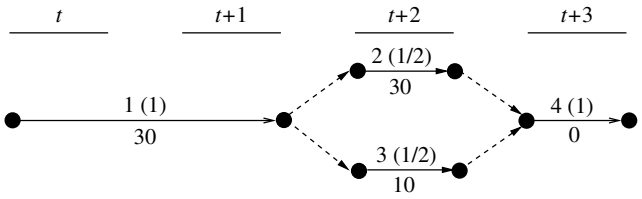
Finally, for any $c \in C^h_{t,\,p+1}$, let $\bar{F}^h_t(c)$ be the polyhedron obtained by adding valid inequality (10) to $\bar{P}^h_t(c)$; we observe that, if an $e \in c$ exists such that $l_e > 1$, (10) is necessary to give the description of $\bar{F}^h_t(c)$.

$\bar{F}^h_t(c)$ has the same dimension as $\bar{P}^h_t(c)$. In fact, for each ε such that $0 < \varepsilon < \min\{p/(p+1), 2(1-1/|c|)\}$, x^h_e defined in Proposition 1 is an interior point also of $\bar{F}^h_t(c)$. To prove that (10) is a facet of $\bar{F}^h_t(c)$, we have to provide $\dim(\bar{F}^h_t(c)) - 1$ linearly independent vectors, which are feasible to $\bar{F}^h_t(c)$ and satisfy (10) as equality. We thus 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 relating 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 $1-1/|c|$. 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 $\bar{F}^h_t(c)$. 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 this matrix there is a block for each service activity in c: an identity block of dimension 2 for each singleton; and an upper triangular block of dimension l_e+1 with nonzero diagonal elements, for each service activity covering $l_e > 1$ days. From the nonsingularity of the matrix, the linear independence of the vectors follows.

Other valid inequalities can be introduced by taking into account the resource constraint (6). W.l.g. we focus on the resource constraint relating to the maximum flight time in a week (time period of length 7). Let d be a positive integer number such that $d \leqslant \min\{7, p\}$, C^h_{td} be defined as in Definition 1, and assume that a feasible subpath $c \in C^h_{td}$ violates the resource constraint. Then, for any $c \in C^h_{td}$, valid inequality (10) 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 is 50. When d and p are fixed at 3, $c = \{1, 2\}$ with $c \in C^h_{t3}$, is a feasible subpath that violates the resource constraint: in fact $30 + 30 > 50$.

Figure 2. An example involving the flight time.



Constraint $x^h_1 + x^h_2 \leqslant 1$, is thus a valid inequality; for example, it cuts off the solution $x^h_1 = 1$, $x^h_2 = x^h_3 = 1/2$, which satisfies the resource constraint because it is $30 + 1/2 \cdot 30 + 1/2 \cdot 10 = 50$.

4. Lifting the Valid Inequalities

The valid inequality (10) can be further tightened; to this end we introduce more notation.

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

t^1_e be the beginning day of activity e, and
t^q_e be the ending day of activity e.

Given a path $c \in C^h_{t,\,p+1}$ covering $p+1$ layers, for each activity arc e belonging to c, $A(e)$ is thus defined as follows:

$$A(e) = \begin{cases} \{k \in A\backslash R \text{ s.t. } l_k \geqslant l_e,\ t^q_e = t^q_k\} \\ \qquad\qquad \text{if } e \text{ is the first activity of subpath } c, \\ \{k \in A\backslash R \text{ s.t. } l_k = l_e,\ t^q_e = t^q_k\} \\ \quad \text{if } e \text{ is an activity in the middle of subpath } c, \\ \{k \in A\backslash R \text{ s.t. } l_k \geqslant l_e,\ t^1_e = t^1_k\} \\ \qquad\qquad \text{if } e \text{ is the last activity of subpath } c. \end{cases}$$

Then, for any $c \in C^h_{t,\,p+1}$, the following valid inequality holds:

$$\sum_{e\in c}\left(x^h_e + \sum_{k\in A(e)} x^h_k\right) \leqslant |c| - 1, \tag{21}$$

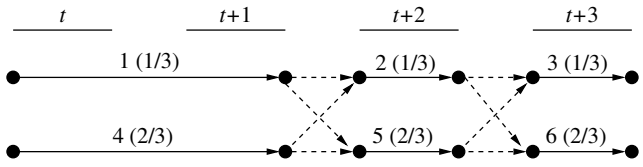
because the constraint

$$x^h_e + x^h_k \leqslant 1 \quad \forall k \in A(e)\ \forall e \in c$$

holds, given that e and k are overlapping activities.

To better understand the valid inequality (21), we consider the example shown in Figure 3. Here there are two rest arcs, namely 2 and 3; and four service activity arcs, 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 of arcs 1, 2, and 3 with a flow equal to $1/3$ and another subpath given by arcs 4, 5, and 6 with a

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



flow equal to 2/3. We have $c = \{4, 5, 6\}$, with $c \in C^h_{t,\,p+1}$, $A(4) = \{1\}$, $A(5) = A(6) = \varnothing$: as a matter of fact activities 1 and 4 have the same length and cover the same time period. The valid inequality $x^h_4 + x^h_5 + x^h_6 \leqslant 2$ does not cut off the current fractional solution, while the tightened valid inequality $x^h_1 + x^h_4 + x^h_5 + x^h_6 \leqslant 2$ cuts off the fractional solution because it is $1/3 + 3 \cdot 2/3 > 2$.

It is straightforward to see that the result of Theorem 1 also holds when 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 go back to the situation in Figure 1: $c = \{5, 6\}$, with $c \in C^h_{t4}$, identifies a feasible subpath of length 4. We now assume that $A(5) = \{7\}$ and $A(6) = \varnothing$, i.e., there exists an activity arc, namely arc 7, which covers the same layers as activity arc 5. Let y_3 be the variable corresponding to arc 7.

It is straightforward to see that the following vectors

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

are linearly independent and satisfy constraint $y_1 + y_2 + y_3 \leqslant 1$ 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 six vectors has been shown before for the case when $A(5) = \varnothing$.

It is easy to see that the same considerations hold when singletons are present; likewise it is not restrictive to assume that $A(e)$ contains at most one element for any e in c.

5. Computational Results

We now discuss some computational results obtained by introducing a subset of constraint (10). In fact, it is worth noting that the valid inequalities proposed in the previous section can be arranged into two groups. The first contains all those constraints that can be treated implicitly in the preprocessing phase, while building graph G; the second consists of those constraints that 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 below. 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 $t + 1$ that 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 $x^h_e + x^h_k \leqslant 1$ holds, and we do not build the compatibility arc between arcs e and k. The constraint was thus considered implicitly in the construction of graph G. On the other hand, it is impossible to consider implicitly the valid inequalities involving more than two variables. If the preprocessing phase alone is not powerful enough, a larger set of valid inequalities could be inserted explicitly. Because the number of valid inequality (10) can be huge, some criteria should be identified to dynamically select a good pool of valid inequalities to add.

Eliminating the compatibility arc rather than introducing explicitly the valid inequality (10) has a two-fold advantage: (i) the number of rows in the linear programming (LP) problem does not increase, and (ii) it allows fractional solutions that satisfy the valid inequality to be cut off. For example, in the case of Figure 1, eliminating the arc between the two consecutive activity arcs 5 and 6 allows us to cut off the following feasible solution $x^h_1 = x^h_2 = 0$, $x^h_3 = x^h_4 = x^h_5 = x^h_6 = 1/2$, which is not cut off by constraint $x^h_5 + x^h_6 \leqslant 1$.

We tested our approach on seven real instances from a medium size Italian airline which are described in Table 1. For these instances the planning horizon is a month (i.e., $m = 30$ or 31) and the maximum no-break period length p is equal to 6.

The problem name, reported in the first column, contains information on the month in which the work assignment is to be done. Then, for each problem, the table gives the number of employees involved, the number of pairings and reserve blocks that should be covered, the number of preassigned activities to be guaranteed, and the maximum length of a pairing (expressed in working days). The last two columns show two trivial upper bounds on the number of activities which can be covered and on their total length, respectively. 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 constraint (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 for our approach.

Tables 2 and 3 give some results obtained with the commercial LP solver CPLEX 7.0 (ILOG 2000). All tests were

Table 1. The data set.

Problem name	Employees	Pairings	Reserves	Preassigned	dMax	z^n_{UB}	z^l_{UB}
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

carried out on an Athlon at 1,200 MHz with 512 MB 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 ($\overline{\text{ACR}}$) and of the problem obtained after the preprocessing phase ($\overline{\text{ACR} + \text{PP}}$) for each objective function chosen, either the max cardinality case or the max length case. In particular, the objective function value, and 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 the activities can be covered, apart from 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 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 the number of columns involved to be reduced significantly even though such a reduction does not always result in a reduction of the computational time required to solve the LP (see Time columns). Hereafter, the computational time reported is given in CPU seconds unless otherwise stated.

In addition, the number of fractional variables in the LP solution, with and without the cuts, is reported (see Frac columns in Tables 2 and 3). The solutions obtained from the LP by adding the cuts are better than the ones obtained without them from an integrality point of view, apart from the first problem in the test bed when the aim is to maximize the number of activities covered. The cuts allow us to reduce the number of fractional variables in the LP solution, no matter 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 considerable 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 was fixed for the most difficult problems in the test bed, and the best solution found when this limit was exceeded is reported. The relative error ε_r is computed as $(z(\text{LP}) - z(\text{IP}))/z(\text{IP})$.

As Tables 4 and 5 clearly show, most of the instances can be solved to optimality within a reasonable computational time, and after that very few nodes of the enumeration tree are explored (the symbol + near the number of nodes indicates that the optimal solution was found via the node heuristic procedure of CPLEX). Moreover, when the preprocessing phase is being carried out, 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. This fact seems to indicate that the model proposed fits well the rostering problem studied.

The validity of our approach, however, becomes clearer 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. Without the cuts,

Table 2. LP results: max cardinality case.

	$\overline{\text{ACR}}$						$\overline{\text{ACR} + \text{PP}}$					
Problem name	Obj	Rows	Columns	Coeff	Frac	Time	Obj	Rows	Columns	Coeff	Frac	Time
Jul0	189	11,229	39,552	107,544	279	77.6	189	11,157	31,368	91,176	347	90.05
Oct0	244	18,993	63,825	179,068	66	55.62	244	17,615	39,816	130,537	36	132.06
Oct1	259	17,486	57,563	162,625	87	53.95	259	16,142	36,183	118,712	40	129.59
Oct2	290	19,941	72,692	195,685	227	120.54	290	19,483	52,003	153,141	92	83.43
Oct3	259	13,088	40,502	117,088	379	25.8	259	12,179	26,739	88,685	44	61.88
Sep0	275	12,878	43,367	122,879	333	111.36	275	12,183	28,248	92,258	308	96.82
Sep1	306	15,026	55,542	150,439	499	198.88	306	14,780	40,352	119,641	384	269.28

Table 3. LP results: max length case.

	$\overline{\text{ACR}}$						$\overline{\text{ACR + PP}}$					
Problem name	Obj	Rows	Columns	Coeff	Frac	Time	Obj	Rows	Columns	Coeff	Frac	Time
Jul0	503	11,229	39,552	107,544	232	53.93	503	11,157	31,368	91,176	205	63.63
Oct0	788	18,993	63,825	179,068	208	192.47	788	17,615	39,816	130,537	12	113.05
Oct1	846	17,486	57,563	162,625	122	44.83	846	16,142	36,183	118,712	28	45.13
Oct2	908	19,941	72,692	195,685	97	71.99	908	19,483	52,003	153,141	0	178.45
Oct3	846	13,088	40,502	117,088	398	26.15	846	12,179	26,739	88,685	114	19.38
Sep0	837	12,878	43,367	122,879	479	80.25	837	12,183	28,248	92,258	319	65.26
Sep1	901	15,026	55,542	150,439	434	222.92	901	14,780	40,352	119,641	297	214.77

CPLEX exceeds the time limit without providing an optimal solution and after two hours only quite a poor feasible solution (relative error over 4%) is reported. For problems Sep0 and Sep1 (see Table 4) as well, CPLEX fails to provide an optimal solution, while with the cuts an optimal solution is reported within 8 and 14 hours, respectively. In any case, we point out 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 were obtained for 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, although better solutions are given, in terms of relative error, than the ones obtained in the max cardinality case. On the other hand, the time required to solve the problem to optimality when the preprocessing phase is enabled is much less than the one required in the max cardinality case: less than two hours for problem Sep0 against the eight hours required by the max cardinality case, and about four hours for Sep1 against 14 hours. Only for the first problem in the test bed was the performance worse: The time required by the max length case increases, in fact, by a factor of 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 also seem to indicate that the max cardinality case outperforms the max length case in terms of the quality of the first feasible solution obtained. We ran a truncated version of the branch-and-bound (B & B) in which CPLEX was stopped as soon as an integer solution was found. Computational results are shown in Tables 6 and 7 for the max cardinality case and the max length case, respectively. When the preprocessing phase is disabled, the max length case provides more accurate solutions than the ones obtained by the max cardinality case, though the time required 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 the 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) did the max cardinality case produce a solution with a relative error bigger than the one obtained for the max length case—requiring, however, half an hour rather than an hour.

6. Conclusions

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 was performed, which consists of taking into account only those valid inequalities that can be treated implicitly in the construction of the compatibility

Table 4. IP results: max cardinality case.

Problem	ACR			ACR + PP		
name	% ε_r	Nodes	Time	% ε_r	Nodes	Time
Jul0	4.42	1,300	15 h	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	13,300	15 h	0	2,941	8 h
Sep1	2.04	2,200	15 h	0	878	14 h

Table 5. IP results: max length case.

Problem	ACR			ACR + PP		
name	% ε_r	Nodes	Time	% ε_r	Nodes	Time
Jul0	3.29	700	15 h	0	143	1 h 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	13,300	15 h	0	239	1 h 46′
Sep1	1.29	3,100	15 h	0.11	240+	4 h 15′

Table 6. Truncated B & B results: max cardinality case.

Problem name	ACR			ACR + PP		
	% ε_r	Nodes	Time	% ε_r	Nodes	Time
Jul0	4.42	178	2 h	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	1 h 30′	0.73	48	30′
Sep1	2.08	397	4 h 20′	0.33	64	1 h 25′

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 were obtained for medium scale problems by enabling the preprocessing phase. As an example, the optimal solution for an instance with 40,352 columns, 14,780 rows, and 119,641 nonzero elements was obtained in about four CPU hours, while without the cuts, CPLEX fails to provide the optimal solution within 15 hours. In any case, quite good solutions (relative error not over 0.7%) were achieved within an hour and an half, when the preprocessing phase is done, for all the real instances solved. We stress the fact that for almost all the test problems, the gap between the upper bound given by the LP relaxation and the optimal integer solution is zero. Generally speaking, this 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. The effectiveness of the preprocessing phase also depends 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 hauls versus the short haul: In the former case, the probability of having covers composed only by two service activities is higher than in the latter case. The preprocessing phase takes advantage of this structure, resulting in a considerable reduction in the computational time required to get the optimal solution. On the other hand, in the short haul, covers are generally composed of more than two activities, and the number of valid inequality (10) that would have to be generated could be huge. For this purpose, 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, to avoid the insertion of the cuts being too burdensome, clever rules need to be devised that 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 to get, hopefully, quite good solutions within reasonable computational times. These are future research issues we would like to pursue.

Table 7. Truncated B & B results: max length case.

Problem name	ACR			ACR + PP		
	% ε_r	Nodes	Time	% ε_r	Nodes	Time
Jul0	3.29	236	5 h 10′	0.6	68	1 h
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	1 h 40′	0.36	79	1 h
Sep1	1.35	273	3 h	1.35	93	1 h 40′

Acknowledgments

This research was supported by MAIOR srl. The authors thank Duccio Grandi, one of MAIOR's managers, for his valuable assistance and for his contribution to the experimentation phase of the work. They are also grateful to the referees and the area and associate editors, whose criticisms and comments helped the authors to improve the quality of this paper.

References

Byrne, J. 1988. A preferential bidding system for technical aircrew. *AGIFORS Sympos. Proc.* **28** 87–99.

Carraresi, P., G. Gallo. 1984. A multi-level bottleneck assignment approach to the bus drivers' rostering problem. *Eur. J. Oper. Res.* **16** 163–173.

Gamache, M., F. Soumis. 1998. A method for optimally solving the rostering problem. G. Yu, ed. *Operations Research in the Airline Industry*. Kluwer, Boston, MA, 124–157.

Gamache, M., F. Soumis, G. Marquis, J. Desrosiers. 1999. A column generation approach for large-scale aircrew rostering problems. *Oper. Res.* **47**(2) 247–263.

Gamache, M., F. Soumis, D. Villeneuve, J. Desrosiers, É. Gélinas. 1998. The preferential bidding system at Air Canada. *Transportation Sci.* **32**(3) 246–255.

Giafferri, C., J. P. Hamon, J. G. Lengline. 1982. Automatic monthly assignment of medium-haul cabin crew. *AGIFORS Sympos. Proc.* **22** 69–95.

Glanert, W. 1984. A timetable approach to the assignment of pilots to rotations. *AGIFORS Sympos. Proc.* **24** 369–391.

ILOG. 2000. *ILOG Cplex* 7.0. User's Manual. ILOG, Mountain View, CA.

Kohl, N., S. E. Karisch. 2004. Airline crew rostering: Problem types, modeling, and optimization. *Ann. Oper. Res.* **127** 223–257. Available at www.carmen.se/research_development/research_reports.htm.

Marchettini, F. 1980. Automatic monthly cabin crew rostering procedure. *AGIFORS Sympos. Proc.* **20** 23–59.

Moore, R., J. Evans, H. Ngo. 1978. Computerized tailored blocking. *AGIFORS Sympos. Proc.* **18** 343–361.

Nemhauser, G. L., L. A. Wolsey. 1988. *Integer and Combinatorial Optimization*. John Wiley and Sons, New York.

Ryan, D. M. 1992. The solution of massive generalized set partitioning problems in air crew rostering. *J. Oper. Res. Soc.* **43** 459–467.

Sarra, D. 1988. The automatic assignment model. *AGIFORS Sympos. Proc.* **28** 23–37.