Università degli Studi di Bologna

Dipartimento di Elettronica Informatica e Sistemistica
viale Risorgimento, 2 40136 - Bologna (Italy)
phone : + 39-51-6443001 fax : + 39-51-6443073

Modeling and Solving the Crew Rostering Problem

Alberto Caprara, Matteo Fischetti,

Paolo Toth and Daniele Vigo

Technical Report DEIS - OR - 95 - 6(R)

Modeling and Solving the Crew Rostering Problem

Alberto Caprara*, Matteo Fischetti•,

Paolo Toth* and Daniele Vigo*

* *DEIS, University of Bologna, Italy*

• *DMI, University of Udine, Italy*

Abstract

The *Crew Rostering Problem* (CRP) aims at determining an optimal sequencing of a given set of duties into rosters satisfying operational constraints deriving from union contract and company regulations. Previous work on CRP addresses mainly urban mass-transit systems, in which the minimum number of crews to perform the duties can easily be determined, and the objective is to evenly distribute the workload among the crews. In typical railway applications, however, the roster construction has to take into account more involved sequencing rules, and the main objective is the minimization of the number of crews needed to perform the duties. In this paper we propose a basic model for CRP, and describe a Lagrangian lower bound based on the solution of an assignment problem on a suitably-defined graph. The information obtained through the lower bound computation is used to drive an effective algorithm for finding a tight approximate solution to the problem. Computational results for real-world instances from railway applications, involving up to 1,000 duties, are presented, showing that the proposed approach yields, within short computing time, lower and upper bound values that are typically very close. The code based on the approach we propose won the FARO competition organized by the Italian railway company, Ferrovie dello Stato SpA, in 1995.

Key Words: crew rostering, Lagrangian relaxation, assignment problem, heuristic algorithm.

A typical problem arising in the management of large transit systems is the following: given a set of *trips* to be covered every day in a given period and a set of *crews*, build a daily assignment of each trip to a crew so as to guarantee that all the trips are covered in the period and the corresponding overall cost is minimized. A widely-used approach to solve this problem consists of decomposing it into two phases. In the *crew scheduling* phase, the short-term schedule of the crews is considered: a convenient set of *duties* is

constructed, each representing a set of trips to be covered by a single crew within a given time period (typically, 24-48 hours). Generally this problem is solved by generating a very large number of potential duties, each with a given cost, and by solving a *Set Covering Problem* (SCP) in order to select a minimum-cost set of duties covering all the trips. In the *crew rostering* phase, a set of *working rosters* is constructed which determine the sequence of duties that each single crew has to perform over the given time period, so as to cover every day all the duties selected in the first phase.

In this paper we focus on the second phase, and address the associated *Crew Rostering Problem* (CRP). Most of the previous works on CRP refer to urban mass-transit systems, where the minimum number of crews to perform the duties can easily be determined, and the objective is to evenly distribute the workload among the crews; see Jachnik (1981), Bodin et al. (1983), Carraresi and Gallo (1984), Hagberg (1985), and Bianco et al. (1992). Set partitioning approaches for airline crew rostering are described in Ryan (1992), Gamache and Soumis (1993), Gamache et al. (1994), and Jarrah and Diamond (1995). Finally, related cyclic scheduling problems are addressed in Tien and Kamiyama (1982), and Balakrishnan and Wong (1990).

In 1994 the Italian Railway Company, *Ferrovie dello Stato SpA*, jointly with *AIRO*, the Italian Operational Research Society, organized two competitions among departments of the Italian universities, in order to promote the design of effective heuristic codes for the crew scheduling and rostering phases. The first competition, named FASTER (Ferrovie Airo Set covering TendER), required the design of algorithms for very-large scale SCP's, involving up to $5,000$ rows and $1,000,000$ columns; see Caprara, Fischetti and Toth (1995). The second competition, named FARO (Ferrovie Airo Rostering Optimization), called for algorithms for the CRP arising in the construction of rosters for railway crews, which is characterized by several operational constraints. The problem objective function was of a hierarchical type, the most important goal being the minimization of the number of crews needed to perform the duties; see AIRO, Ferrovie dello Stato SpA (1994). Each participant to the FARO competition had to design a code to be sent to Ferrovie dello Stato SpA. Two prizes of approximately US$ 60,000 and US$ 30,000 were to be assigned to the codes giving the best and the second best solution values, respectively, on three instances with up to $1,000$ duties, within 90 minutes on a PC 486/33 with 8 Mbyte RAM. We took part in the FARO competition as the unit of the Dipartimento di Elettronica, Informatica e Sistemistica, Università di Bologna. Our code won the first prize, and gave the best solutions for all the instances of the competition; see AIRO, Ferrovie dello Stato

SpA (1995).

In this paper we propose a general model designed for airline/railway rostering applications, and we develop heuristic algorithms for its solution. The paper is organized as follows. Section 1 presents a general description of the problem, and gives graph-theoretical and integer programming formulations. In Section 2 we illustrate a Lagrangian lower bound based on the solution of an assignment problem on a suitably-defined graph. The information obtained through the lower bound computation is used in Section 3 to drive an effective algorithm for finding an approximate solution to the problem. Section 4 gives a detailed description of the FARO competition problems. Computational results are presented in Section 5, showing that the proposed approach yields, within a short computing time, lower and upper bound values that are typically very close.

1 A general model

In this section we present a model for the class of rostering problems we consider, which includes some main features of the airline/railway applications. The formulation can easily be extended to take into account other problem-specific constraints which can arise in practical situations, see Section 4. Unless explicitly specified, all times are integer and are expressed in minutes.

The problem we consider is periodic, in the sense that each duty has to be covered every day. (Situations in which there are slight differences in the workload of some days, e.g., Sundays, are typically dealt with by heuristically rearranging the solution associated with a periodic basic problem.)

We are given a set of n *duties* to be covered by a set of *crew rosters*. Each duty i has a *start time*, s_i, $0 \leq s_i < 1440$ (= 24 hours), and a *duration* p_i. Moreover, each duty i has an associated *working time*, $w_i \leq p_i$, which is the time actually spent working during the duty, and a *paid time*, $a_i \geq w_i$, which is the sum of the working time and all the possible additional paid time intervals of the duty (e.g., short rests and transfers). We allow $a_i > p_i$, as in practice some of the paid time intervals can be fictitious. Each duty can have additional *attributes*, which are not all mutually exclusive and are explicitly given on input. For example, a duty can be an *overnight duty* if it requires working during the night, or a *long duty* if its working time is greater than a given threshold, etc.

A *week* is conventionally defined as a group of k consecutive days. A *roster* consists of a subset of duties, and spans over a cyclic sequence of consecutive weeks. The *length* of a roster is defined as the number of its days (an integer multiple of k). Typically, an upper

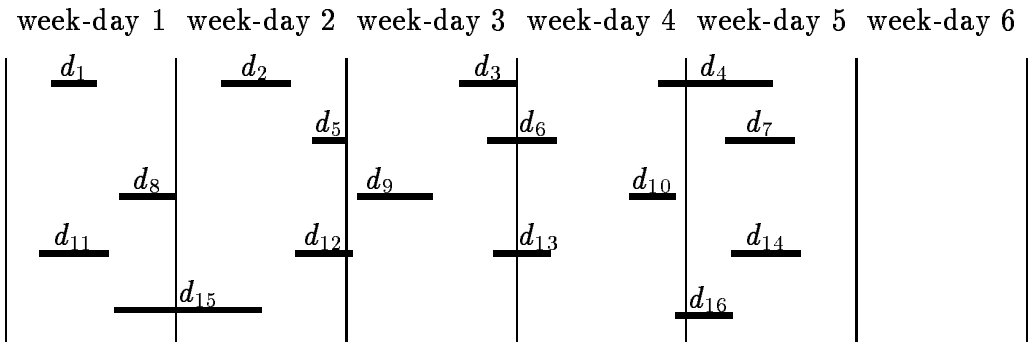

Figure 1: An example of a roster

bound q on the length of the roster is imposed. The periodic nature of the problem implies that the length of a roster gives the number of crews needed to cover its duties every day. We call *complete day* a time interval of 24 hours (i.e., 1440 minutes) starting at midnight. Moreover, a complete day is called *idle* if no duty or part of a duty is executed during that day, otherwise the day is called *working*. We also define

$$\alpha = k \cdot 1440 = \text{number of minutes in a week.} \tag{1}$$

An example of a roster is illustrated in Figure 1. Here a week spans $k = 6$ days. The roster consists of a cyclic duty sequence $d_1, \ldots, d_{16}, d_1, \ldots$, and spans 5 weeks in which each 6-th day is idle. The roster length is then 30 days. Accordingly, 30 crews are needed to perform each daily occurrence of $d_1, \ldots, d_{16}$. Indeed, a first crew covers: on *calendar* day x, say, duty d_1, on day $x + 1$ duty d_2, ..., on day $x + 29$ no duty, on day $x + 30$ duty d_1 again, and so on. On day $x + 1$, duty d_1 is instead covered by a second crew, which also performs on day $x + 2$ duty d_2, and so on. Analogously, duty d_1 on day $x + 2$ is covered by crew number 3, on day $x + 3$ by crew number 4, and finally on day $x + 29$ by crew number 30. In other words, on each calendar day, the 30 crews perform the assignments of a different day of the roster.

Clearly, feasible rosters have to include *weekly rests* for the crews, which may be of different types. Although the model does not require this, for the sake of concreteness we consider a common situation arising in railway applications, where each week in a roster is separated by the next one through a continuous rest, which always spans the complete k-th day of the week, i.e., every k-th day is *idle*. There are two types of weekly rest, conventionally called *simple* and *double* weekly rests, each characterized by a different

minimum length. Double weekly rests are generally longer than simple ones; in each roster their number must be at least equal to a given fraction $\beta > 0$ of the total number of weeks in the roster. The average length of the weekly rests, computed over all the weeks in a roster, must be at least equal to a given threshold. Moreover for a given attribute j, for each week and for each cyclic group of m_j consecutive days, an upper bound is imposed on the total number of duties having attribute j. Similar upper bounds are also imposed on the total working and paid time of the duties. In the example in Figure 1, no more than 2 overnight duties can be included in each week, and no more than 7 overnight duties can be included in each cyclic group of 30 consecutive days. In addition, the total working time cannot exceed 36 hours for each cyclic group of 7 consecutive days, and the total paid time cannot exceed 170 hours for each cyclic group of 30 consecutive days.

Two consecutive duties of a roster, say i and j, can be sequenced either *directly* (without an intermediate weekly rest), or with a simple or double weekly rest between them. For each type of sequencing and for each set of attributes of duties i and j, a minimum time interval between the end of duty i and the start of duty j is imposed. In the example in Figure 1, the minimum time interval between two duties i and j sequenced directly is 22 hours if they are both overnight, and 18 hours otherwise. This explains why, for instance, duties d_2 and d_3 are not both scheduled on the second day of the roster, and why an idle day (the 14-th day in the roster) is present between duties d_8 and d_9. For simple and double weekly rests, instead, the minimum time interval is 48 hours and 2 idle days, respectively. For instance, the first and the third weekly rests are double, while all the others are simple. All the constraints concerning the sequencing of two consecutive duties within a roster (independently of the other duties in the roster) will be called *sequencing rules*, while all the remaining constraints imposed on CRP will be called *operational constraints*. Notice that operational constraints may require the insertion of additional idle days between two consecutive duties (see duties d_{15} and d_{16} in Figure 1).

Problem CRP then consists of finding a feasible set of rosters, covering all the duties and minimizing the total number of weeks in the rosters. As already observed, the global number of crews required every day to cover all duties is equal to k times the total number of weeks. Thus the minimization of the number of weeks implies the minimization of the global number of crews required.

In the following we give a formulation of CRP as a graph-theoretical problem. We are given a complete directed multigraph $G = (V, A \cup L)$, where $V = \{1, \ldots, n\}$ is the set of vertices, A is a set of arcs, and L is a set of loops. Each vertex is associated with a duty.

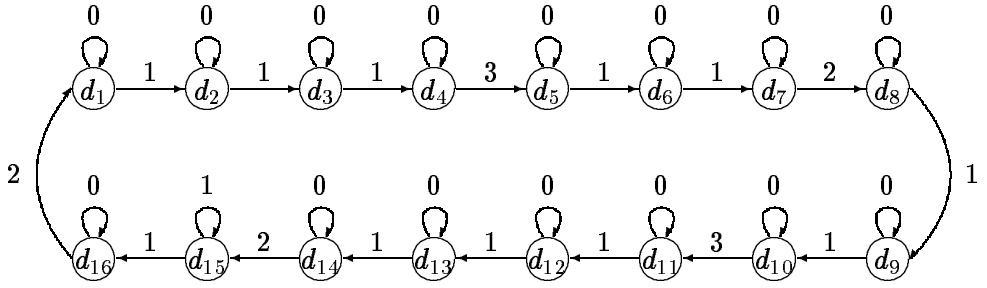

Figure 2: A looped cycle corresponding to a feasible roster.

The arcs from vertex i to vertex j represent the consecutive sequencing of duty pair i, j within a roster, while each loop incident with a vertex i represents idle days spent between the end of duty i and the start of the subsequent duty in the roster. To be more specific, the arc set A contains arcs of three different types and is partitioned into three subsets, A_1, A_2 and A_3. Arcs belonging to A_1 are called *direct arcs*, while arcs belonging to A_2 (resp., A_3) are called *simple rest arcs* (resp., *double rest arcs*). For each pair of vertices $i, j \in V$ we have a direct arc $(i, j) \in A_1$, whose length c^1_{ij} is the minimum time, in minutes, between the start of duty i and the start of duty j when they are sequenced directly, i.e., in the same week. In other words, $c^1_{ij} := (s_j + h \cdot 1440) - s_i$, where h is the minimum number of complete days leading to a feasible direct sequencing. Similarly, we have a simple rest arc $(i, j) \in A_2$ (resp., a double rest arc $(i, j) \in A_3$) whose length c^2_{ij} (resp., c^3_{ij}) is the minimum time, in minutes, between the start of duty i and the start of duty j when a simple (resp., double) weekly rest is imposed between them. Matrices c^1, c^2 and c^3 can easily be computed from the input data, according to the sequencing rules. For all i we also define $c^1_{ii} := c^2_{ii} := +\infty$, and $c^3_{ii} := \alpha$ (as defined in (1)), so as to take care of 1-duty rosters. Notice that, by definition, for any given pair $i, j \in V$ the values $c^1_{ij}, c^2_{ij}, c^3_{ij}$ differ by integer multiples of 1440. The loop set L is partitioned into $\sigma + 1$ subsets, $L_0, \ldots, L_\sigma$, where σ is an upper bound on the number of idle days between two duties. For each vertex $i \in V$ and for $t = 0, \ldots, \sigma$, we have a loop $(i, i) \in L_t$ of length $d^t_i := t \cdot 1440$, representing a rest of t idle days between i and the subsequent duty in the roster.

A *looped path* is a (possibly closed) path of G of the form $P = \{(v_1, v_1), (v_1, v_2), (v_2, v_2), (v_2, v_3), \ldots, (v_l, v_l), (v_l, v_{l+1})\}$, where $v_1, \ldots, v_{l+1}$ are distinct vertices (with $v_{l+1} = v_1$ if the path is closed), i.e., a simple path amended by a series of loops, one for each vertex of the path except the last one. A *looped cycle* is a closed looped path. Each feasible roster

then corresponds to a looped cycle of G (note that the converse does not hold). As an illustration, Figure 2 shows the looped cycle corresponding to the roster in Figure 1. For each arc $(i,j) \in A_t$ $(t = 1,2,3)$ or loop $(i,i) \in L_t$ $(t = 0,\ldots,\sigma)$ belonging to the solution, we report in the figure the label "t" besides the corresponding arc/loop. Notice that the idle day after duty d_8 follows from the sequencing rules, hence loop $(d_8,d_8) \in L_0$, while the idle day after duty d_{15} follows from the operational constraints, hence loop $(d_{15},d_{15}) \in L_1$.

An equivalent graph-theoretical model, which avoids the use of loops, could be obtained by using the following standard transformation:

- each vertex $i \in V$ is split into two copies i^- and i^+;
- each arc $(i,j) \in A$ is replaced by an arc (i^+,j^-);
- each loop $(i,i) \in L$ is replaced by an arc (i^-,i^+).

In this way looped paths of the original graph correspond to simple paths of the transformed graph.

In order to derive a mathematical formulation for CRP, we introduce the notion of *infeasible looped path*, defined as a looped path which cannot be contained in any looped cycle corresponding to a feasible roster. An infeasible looped path is called *minimal* if it does not contain another infeasible looped path.

CRP then calls for the determination of a minimum-length set of disjoint looped cycles of G containing no infeasible looped path, and such that each vertex is covered by exactly one cycle. It is worth noting that the feasibility of the overall solution follows from that of each single roster.

We next give an *Integer Linear Programming* (ILP) model for CRP, based on the above-described graph-theoretical formulation. In the model the sequencing rules are implicitly imposed by means of the length matrices c^1, c^2, c^3, whereas the operational constraints are modeled via inequalities which forbid the occurrence of infeasible looped paths.

For each arc $(i,j) \in A_t$, $i,j = 1,\ldots,n$, $t = 1,2,3$, we introduce a binary variable x_{ij}^t equal to 1 if arc $(i,j) \in A_t$ is in the optimal solution, and 0 otherwise. Similarly, for each loop $(i,i) \in L_t$, $i = 1,\ldots,n$, $t = 0,\ldots,\sigma$, we introduce a binary variable y_i^t equal to 1 if loop $(i,i) \in L_t$ is in the optimal solution, and 0 otherwise. Finally, we let $\mathcal{P}$ be the set of all minimal infeasible looped paths of G. The model then reads

$$v(\text{CRP}) = \min \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{t=1}^{3} c_{ij}^t x_{ij}^t + \sum_{i=1}^{n}\sum_{t=0}^{\sigma} d_i^t y_i^t \qquad (2)$$

subject to

$$\sum_{i=1}^{n}\sum_{t=1}^{3} x_{ij}^{t} = 1, \quad j = 1,\ldots,n \tag{3}$$

$$\sum_{j=1}^{n}\sum_{t=1}^{3} x_{ij}^{t} = 1, \quad i = 1,\ldots,n \tag{4}$$

$$\sum_{t=0}^{\sigma} y_{i}^{t} = 1, \quad i = 1,\ldots,n \tag{5}$$

$$\sum_{t=1}^{3}\sum_{(i,j)\in A_t\cap P}(1-x_{ij}^{t}) + \sum_{t=0}^{\sigma}\sum_{(i,i)\in L_t\cap P}(1-y_{i}^{t}) \geq 1, \quad P\in\mathcal{P} \tag{6}$$

$$x_{ij}^{t}\in\{0,1\}, \quad i,j = 1,\ldots,n;\ t = 1,2,3 \tag{7}$$

$$y_{i}^{t}\in\{0,1\}, \quad i,j = 1,\ldots,n;\ t = 0,\ldots,\sigma. \tag{8}$$

Constraints (3) and (4) impose that each vertex has exactly one in-going arc and one out-going arc, respectively, while constraints (5) ensure the solution contains exactly one loop incident with each vertex.

The presence of infeasible looped paths P in the solution is forbidden by constraints (6), which stipulate that at least one variable associated with the arcs/loops of P must be set to 0. These constraints could be reinforced in several ways, so as to produce tighter LP relaxations. We do not pursue this objective in the present paper, since we are interested in more combinatorial relaxations.

Note that the length of any looped cycle is an integer multiple of 1440, hence the objective function value corresponds to an integer number of days, namely the total number of days required to cover all the duties.

Our model can easily be modified in order to take into account many possible variations of the problem. Among others, we mention the case in which the number of possible types of sequencing of subsequent duties is, say, q. This can be handled by introducing q different

arcs between each duty pair in the graph G. Another possible variation arises when the weeks in a roster can have different lengths, provided a weekly rest is scheduled at the end of each week, and each roster contains a number of days, say r, which is multiple of k, and r/k weekly rests. This latter situation does not require any change in the above formulation. Other examples of additional features that are easily included in the model are given in Section 4.

We now extend the ILP model above by adding some additional variables and inequalities which are redundant as long as the infeasible-path constraints (6) are imposed, but turn out to be useful in the relaxation of CRP defined in the next section, where these constraints are removed. The new variables and inequalities impose that the total number of weekly rests is at least equal to the total number of weeks composing the rosters, and that the total number of double weekly rests is at least equal to β times the total number of weekly rests. In our graph-theoretical formulation, these constraints can be stated as:

i) the total number of simple or double rest arcs in the solution has to be at least equal to the total length of the cycles, expressed in weeks;

ii) the total number of double rest arcs in the solution cannot be less than β times the total number of simple or double rest arcs.

These constraints are modeled by introducing two integer variables: w, representing the minimum number of simple or double rest arcs in the solution, and z, representing the minimum number of double rest arcs in the solution. The new inequalities associated with *i*) and *ii*) then read:

$$w \geq \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{t=1}^{3} c_{ij}^{t} x_{ij}^{t} + \sum_{i=1}^{n}\sum_{t=0}^{\sigma} d_{i}^{t} y_{i}^{t}}{\alpha} \tag{9}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n}(x_{ij}^{2} + x_{ij}^{3}) \geq w \tag{10}$$

$$z \geq \beta w \tag{11}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij}^{3} \geq z \tag{12}$$

$$w, z \geq 0 \ \text{ integer.} \tag{13}$$

2 Lower bounds

Simple lower bounds on the length (in minutes) of a CRP solution can easily be obtained in $O(n)$ time, by considering each of the constraints imposing a limit on the total number of duties with a given attribute j, or on the total working and paid time, in a week and in a cyclic group of m_j consecutive days in a roster. For instance, if the total paid time cannot exceed 170 hours for each cyclic group of 30 consecutive days, a lower bound can be obtained as $(1440 \cdot 30) \sum_{i=1}^{n} a_i/(170 \cdot 60)$. Bounds of this type will be called *trivial bounds* in the sequel.

We now describe a more sophisticated relaxation of CRP, derived from the ILP formulation introduced in the previous section. The relaxed problem is defined as follows. First, we remove the infeasible looped path constraints (6). This allows one to get rid of the loop variables y_i^t, as in any optimal solution one has $y_i^0 = 1$ and $y_i^1 = \ldots = y_i^\sigma = 0$ for all i. Then, we relax constraints (10) and (12) in a Lagrangian way, using nonnegative Lagrangian multipliers λ_1 and λ_2, respectively, and obtain the objective function

$$\min \quad \lambda_1 w + \lambda_2 z + \sum_{t=1}^{3} \sum_{i=1}^{n} \sum_{j=1}^{n} \overline{c}_{ij}^t x_{ij}^t, \tag{14}$$

where for all $i, j = 1, \ldots, n$ the values $\overline{c}_{ij}^1 := c_{ij}^1$, $\overline{c}_{ij}^2 := c_{ij}^2 - \lambda_1$ and $\overline{c}_{ij}^3 := c_{ij}^3 - \lambda_1 - \lambda_2$, are the *Lagrangian costs* for the variables x_{ij}^1, x_{ij}^2 and x_{ij}^3, respectively. Finally, we replace constraint (9) with the inequality

$$w \geq \frac{\lambda_1 w + \lambda_2 z + \sum_{t=1}^{3} \sum_{i=1}^{n} \sum_{j=1}^{n} \overline{c}_{ij}^t x_{ij}^t}{\alpha}, \tag{15}$$

whose validity follows from the easy observation that, for all feasible solutions, the right-hand-side value in (15) can never exceed that in (9). Let LRP(λ_1, λ_2) denote problem (14), (3), (4), (15), (11), (7), (8) and (13), and let $v(\text{LRP}(\lambda_1, \lambda_2))$ be its optimal solution value. We remark that this relaxed problem takes into account *all* the sequencing rules. The following proposition shows that $v(\text{LRP}(\lambda_1, \lambda_2))$ can be computed by solving an associated *Assignment Problem* (AP).

Proposition 1 *For any given pair of multipliers* $\lambda_1, \lambda_2 \geq 0$*, an optimal solution to LRP*(λ_1, λ_2) *can be computed by:*

i) solving the AP on the cost matrix defined by $\gamma_{ij} := \min\{\overline{c}^1_{ij}, \overline{c}^2_{ij}, \overline{c}^3_{ij}\}$ *for* $i, j = 1, \ldots, n$*, thus obtaining the solution value* $v(\mathrm{AP})$*;*

ii) determining the minimum value w *such that*

$$\alpha w \geq \lambda_1 w + \lambda_2 \lceil \beta w \rceil + v(\mathrm{AP}),\ w \geq 0 \text{ integer}; \tag{16}$$

iii) defining $z := \lceil \beta w \rceil$*.*

Proof. In order to optimize the objective function in $\mathrm{LRP}(\lambda_1, \lambda_2)$, one is first interested in determining the smallest possible value for

$$\sum_{t=1}^{3} \sum_{i=1}^{n} \sum_{j=1}^{n} \overline{c}^t_{ij} x^t_{ij} \tag{17}$$

subject to (3), (4), (7) and (8). This task amounts to finding a minimum Lagrangian-cost set of disjoint cycles of G such that each vertex is covered by exactly one cycle. Clearly, for each vertex pair $i, j \in V$ one can keep only the arc having the minimum Lagrangian cost among the three arcs $(i, j) \in A_t$ $(t = 1, 2, 3)$, and solve the AP on the resulting cost matrix (γ_{ij}). Once the optimal solution value $v(\mathrm{AP})$ has been computed, the minimum (and best) possible value for w is given by (16), which is derived from (15) by replacing z with $\lceil \beta w \rceil$, i.e., by the minimum value allowed by (11). □

Accordingly, solving $\mathrm{LRP}(\lambda_1, \lambda_2)$ essentially amounts to solving an AP problem, which takes $O(n^3)$ time in the worst case.

In order to get the best posible Lagrangian lower bound, one is interested in finding multipliers $\lambda^*_1, \lambda^*_2 \geq 0$ which maximize $v(\mathrm{LRP}(\lambda_1, \lambda_2))$. This can be done by using standard iterative techniques, such as subgradient optimization.

Even if AP's can be solved fairly quickly in practice, the subgradient procedure could be rather time consuming for large-size instances. Furthermore, computational experience has shown that for the real-world instances we consider, the optimal values for λ_1, λ_2 are usually $\lambda^*_1 = \lambda^*_2 = 1440$, or, in a few cases, $\lambda^*_1 = 1440$, $\lambda^*_2 = 0$. This is related to the structure of the arc lengths, which are such that, for any vertex pair i, j, the values $c^1_{ij}, c^2_{ij}, c^3_{ij}$ differ by integer multiples of 1440. Moreover, one typically has $c^3_{ij} = c^2_{ij} + 1440$, although case $c^3_{ij} = c^2_{ij}$ is also possible. We then compute our Lagrangian lower bound as

$LB := \max\{v(\text{LRP}(1440,1440)),\ v(\text{LRP}(1440,0))\}$, by solving only two AP's. The value LB is expressed in minutes; due to the structure of the arc lengths and to the values we assign to the multipliers, it always corresponds to an integer number of days, namely $d := LB/1440$. In addition, the optimal value of variable w gives a typically very tight lower bound on the number of weeks in an optimal solution.

We define a global lower bound LB^* as the maximum among LB and the trivial bounds mentioned at the beginning of this section.

3 The heuristic algorithm

In this section we describe a constructive heuristic for CRP, which extensively uses the information obtained from the solution of the relaxed problem defined in the previous section. The heuristic constructs one roster at a time, choosing in turn the duties to be sequenced consecutively in the roster. Once a roster has been completed, all the duties it contains are removed from the problem. The process is iterated on the remaining duties until all duties have been sequenced. We next outline the procedure we use to build each single roster, as it applies to the construction of the first roster.

Let (u,v) be an optimal dual solution to the AP corresponding to the best Lagrangian lower bound LB, where v_i (resp., u_i) is the optimal dual variable associated with the i-th constraint of type (3) (resp., of type (4)), for $i = 1,\ldots,n$. For each pair $i,j \in V$ and for $t = 1,2,3$, the reduced cost of arc $(i,j) \in A_t$ is $\tilde{c}^t_{ij} := \overline{c}^t_{ij} - u_i - v_j \geq 0$, representing a lower bound on the increase of objective function $v(\text{AP})$ if arc $(i,j) \in A_t$ is imposed in the solution. According to our experience, $\tilde{c}^t_{ij}$ gives a much more accurate estimate than the original length c^t_{ij} of the likelihood of arc $(i,j) \in A_t$ to be in an optimal solution. One is therefore interested in possibly constructing a collection of rosters where only zero reduced-cost arcs are used.

We start building the roster by selecting its initial duty, say i_0, which will be performed at the beginning of a week, i.e., just after a weekly rest. Once the initial duty has been selected, a sequence of iterations is performed where:

a) the best duty to be sequenced after the last duty in the current roster is chosen;

b) the Lagrangian lower bound LB and the trivial bounds are parametrically updated, in $O(n^2)$ time;

c) the possibility of "closing" the roster is considered, possibly updating the current

best roster.

The procedure is iterated until no better roster than the current best one can be constructed, stopping anyway if the roster length attains its maximum value (q days). At the end of the procedure the current best roster is added to the current overall CRP solution, and a new roster is constructed if some duties are still not covered.

The next subsections give a more detailed description of the steps of the roster construction procedure. As a general rule, the algorithm mainly tries to minimize the number of days in the solution; to this end, no idle day is left between two consecutive duties, unless strictly necessary.

3.1 Choice of the initial duty

Duty i_0 is chosen as the duty i having the best value of a score which takes into account the number of arcs with zero reduced cost which are incident with vertex i. In particular, since the initial duty is scheduled at the beginning of a week (generally on the first day), we give priority to the duties having a small number of zero reduced-cost in-going direct arcs (i.e., arcs belonging to A_1), and a large number of zero reduced-cost in-going simple and double rest arcs (i.e., arcs belonging to A_2 and A_3). Moreover, since the initial duty is likely to be followed by other duties in the same week, we strongly penalize duties which have no zero reduced-cost out-going direct arc.

3.2 Choice of the next duty

We choose the duty j to be sequenced after the current duty i (i.e., the last duty in the current roster) as follows. For each candidate duty h we consider the sequencing of h after i through three possible *moves*: direct move μ_h^1, simple weekly rest move μ_h^2, and double weekly rest move μ_h^3, corresponding to the arc (i,h) belonging to A_1, A_2, and A_3, respectively. For each move μ_h^l $(l = 1,2,3)$ we schedule duty h at the earliest time for which all the constraints are satisfied. Move μ_h^l is assigned a score τ_h^l taking into account the increase θ_h^l of the global lower bound value LB^* when arc $(i,h) \in A_l$ is imposed in the solution. In the computation of θ_h^l it is necessary to consider the number δ_h^l of additional idle days (with respect to c_{ih}^l) to be inserted between i and h for the move to be feasible (see Section 3.3). In addition, score τ_h^2 (resp., τ_h^3) is penalized if the number of double weekly rests already performed in the roster is smaller (resp., greater) than β times the current number of (possibly incomplete) weeks included in the roster.

Additional terms are present in the scores, in order to break ties. These terms take into account both the number of arcs with zero reduced cost entering and leaving node h, and the attributes of duty h. The aim is sequencing first the duties which are "critical", i.e., duties having a small number of in-going and out-going arcs with zero reduced cost, large working or paid time, or some additional attributes (overnight, etc.). The weight of each term in the score is dynamically updated so as to consider the tightness of the associated constraint, evaluated as a function of the ratio between the corresponding trivial bound and the global lower bound LB^*. Finally, the scores try to evenly distribute the workload among rosters and among weeks inside a roster. Duty h is then assigned a score $\tau_h := \min\{\tau_h^1, \tau_h^2, \tau_h^3\}$, and the duty j to be sequenced after duty i is chosen as the one having the minimum score.

The computation of values θ_h^l for all duties h can be performed in $O(n^2)$ time by using AP parametric techniques for the computation of the Lagrangian bound. Thus, since both the feasibility check and the computation of the additional terms can be carried out in constant time for each duty h, the overall time complexity of this step is $O(n^2)$.

When few duties (say less than 100) remain to be sequenced, the choice of j is more accurate, using a more time-consuming look-ahead technique. In particular, we add to each score τ_h^l $(l = 1, 2, 3)$ the minimum score corresponding to the sequencing of any other duty g after move μ_h^l. These additional scores are computed analogously to the previous ones, but here we estimate the Lagrangian lower bound increase as the reduced cost $\tilde{c}_{hg}^l$ plus δ_g^l. The above minimum score can be computed in $O(n)$ time for each duty h, hence the overall time complexity of this step remains $O(n^2)$.

3.3 Lower bound recomputation

After the insertion of each duty, we compute the new lower bound value LB^*. As to the Lagrangian lower bound LB, we consider a modified CRP where, for each pair of duties that have been consecutively sequenced so far, the corresponding arc of G is imposed in the solution. This is simply done by setting to $+\infty$ the length of all the other arcs joining sequenced duties, and by increasing LB by the additional idle days possibly included in the current roster. The new AP's required to compute LB are solved parametrically.

3.4 Closing the roster

Given the current duty i, for each duty j that can be feasibly sequenced after i we consider the possibility of closing the roster right after j, i.e., of sequencing duty j as soon as possible

after i (with a move μ_j^l), and then the initial duty i_0 preceded by a simple (or double, if necessary) weekly rest. This requires checking of the feasibility of the resulting roster. If the roster is feasible, it is assigned a score taking into account the difference between the values of lower bound LB^* before and after the construction of the roster. The Lagrangian lower bound LB after the construction of the roster is estimated as follows. Let $g = 3$ if a double weekly rest between j and i_0 is needed for the roster to be feasible, $g = 2$ otherwise. Furthermore, let ϵ be the total number of additional idle days (with respect to the original lengths c_{ij}^l and $c_{ji_0}^g$) which have to be inserted between i and j and between j and i_0 to ensure feasibility. Then, $LB = \overline{LB} + \theta_j^l + \overline{\theta}_j^g + 1440 \cdot \epsilon$, where $\overline{LB}$ is the Lagrangian lower bound computed after the insertion of duty i, θ_j^l is defined as in Section 3.2, and $\overline{\theta}_j^g$ is the increase of the Lagrangian lower bound value when arc $(j, i_0) \in A_g$ is imposed in the solution of CRP. The computation of values θ_j^l and $\overline{\theta}_j^g$ for all duties j can be performed in $O(n^2)$ time by using AP parametric techniques.

The overall complexity of this step is $O(n^2)$.

3.5 Overall heuristic algorithm

Since a roster is not further extended if it is q days long (where q is a fixed value), the overall complexity of our roster construction procedure is $O(n^2)$, from which the overall complexity $O(n^3)$ of our algorithm follows.

When a complete solution to the problem is available, we try to improve it by applying a *refining procedure*. For each roster we compute the difference between the global lower bounds on the original problem, computed with and without imposing the roster in the solution, respectively. We then remove from the solution all the rosters for which the above difference is positive, and re-apply the heuristic algorithm to the corresponding duties. Before this, some parameters of the roster construction procedure are changed, either with a random perturbation, or deterministically in an adaptive fashion, so as to take into account the constraints that made the construction of the removed rosters difficult.

We apply the refining procedure to the best solution obtained, until a given time limit is reached.

4 Application to the Italian railways

In this section we give a detailed description of the real-world CRP proposed by Ferrovie dello Stato SpA within the FARO competition.

In the FARO problem, all the durations and paid times of the duties are not greater than a complete day. Each duty can have the following additional attributes:

- *duty with external rest*, if it includes a rest out of the depot for the crew;
- *long duty*, if it does not include an external rest and its working time w_i is longer than 8 hours and 5 minutes;
- *overnight duty*, if its working period overlaps the interval from midnight to 5:00 am;
- *heavy overnight duty*, if it is an overnight duty without external rest which requires more than 1 hour and 30 minutes of work between midnight and 5:00 am.

A week is a group of 6 consecutive days, i.e., $k = 6$. The length of a roster is typically 30 days (5 weeks) and must not exceed 60 days (10 weeks), i.e., $q = 60$.

Each idle day between two duties in the same week is called a *technical interval*. In the roster in Figure 1, two technical intervals are present (days 14 and 27). If possible, the occurrence of technical intervals is to be avoided.

4.1 Sequencing rules

The minimum rest between the end of a duty and the start of the subsequent duty within a week is 18 hours, unless both duties are overnight. In this case, if at most one of them is heavy overnight, the minimum rest is 22 hours, otherwise the rest must span a complete day. Moreover, operational constraints impose that after two consecutive overnight duties in a week whose intermediate rest does not span a complete day, the rest before the start of any other duty in the same week must be at least 22 hours. The direct sequencing of long duties is never allowed.

Simple weekly rests must be at least 48 hours long, whereas double weekly rests must span at least two complete days, i.e., either the fifth and sixth day of a week or the sixth day of a week and the first day of the subsequent one.

When a simple weekly rest is preceded by an overnight duty, then either the first duty in the next week starts after 6:30 am, or the rest must span two complete days. Note that in this latter case the weekly rest does not necessarily become a double weekly rest (see below). Finally, if the first duty in a week following a double weekly rest starts before 6:00 am, then the rest must span at least three complete days, including the first day of the week (i.e., either the last two days of the previous week and the first day of the current week, or the sixth day of the previous week and the first two days of the current week).

4.2 Operational constraints

Each week can include at most the following number of duties having particular attributes:

- 2 duties with external rest;
- 1 long duty;
- 2 overnight duties.

For each roster the following constraints on the weekly rests are imposed:

- the number of double weekly rests must be at least equal to 40% of the total number of weekly rests, i.e., $\beta = 0.4$;
- the average weekly rest time must be at least equal to 58 hours.

Moreover, for each cyclic group of 30 consecutive days within a roster we have the following constraints:

- no more than 7 duties with external rest can be included;
- the total paid time of the included duties cannot exceed 170 hours.

Finally, for each cyclic group of 7 consecutive days within a roster the total working time of the included duties cannot exceed 36 hours. Notice that each week must have its sixth day idle, hence there is always a group of 7 consecutive days including only the duties in the week, and therefore 36 hours is also an upper bound on the total working time of the duties in a week.

4.3 Special roster

A solution is allowed to include a single *special roster* made up of only one week. In this week the last three complete days must be idle. The only other constraints are the above-defined rules for sequencing duties within the same week (no operational constraint is imposed). Each idle day of the special roster not followed by a duty (with the exception of the last two days) is called *available day*. Notice that the number of available days can be equal to 1, 2 or 3. The occurrence of a special roster with many available days is highly appreciated, since it corresponds to a "soft" roster which can easily be covered by using spare crews.

4.4 Problem objective

The problem calls for finding a feasible set of rosters covering all the duties and optimizing a hierarchical objective function which requires, in decreasing order of importance:

a) the minimization of the total number of weeks making up the rosters of the solution (possibly including the special roster);

b) the minimization of the number of technical intervals in the solution;

c) the maximization of the number of available days (by definition, the number of available days is zero if the special roster is not used).

4.5 Trivial lower bounds

For the computation of the trivial lower bounds, the following five attributes of the duties have been considered: paid time, long duty, overnight, external rest, and working time. According to the operational constraints, the following values represent trivial lower bounds (expressed in minutes) when the special roster is not included in the solution:

- $L_1 = \alpha \cdot 5 \cdot \sum_{i=1}^{n} a_i/(170 \cdot 60)$;
- $L_2 = \alpha \cdot$ number of long duties;
- $L_3 = \alpha \cdot$ (number of overnight duties)/2;
- $L_4 = \alpha \cdot 5 \cdot$ (number of duties with external rest)/7;
- $L_5 = \alpha \cdot \sum_{i=1}^{n} w_i/(36 \cdot 60)$.

When the special roster is imposed in the solution, the trivial lower bounds can be computed by considering each possible number e of available days in the special roster. For each value of e ($e = 1,2,3$) and for each attribute j ($j = 1,\ldots,5$), we assign to the special roster the set of duties maximizing the "load" associated with attribute j (sum of the paid times, ...) for a period of $4 - e$ working days. The corresponding trivial lower bound L'_{je} is given by $1440 \cdot (4 - e)$ plus the bound for attribute j computed with respect to the remaining duties. It follows that a valid trivial lower bound for attribute j is $\tilde{L}_j := \min\{L_j, L'_{j1}, L'_{j2}, L'_{j3}\}$. The upper bound $\tilde{e}_j$ on the number of available days in the solution is given by the maximum value of e such that $\left\lceil L'_{je}/\alpha \right\rceil = \left\lceil \tilde{L}_j/\alpha \right\rceil$, with $\tilde{e}_j := 0$ if no such e exists.

The trivial lower bounds give no information about the number of technical intervals in the optimal solution.

4.6 Lagrangian bound

We now describe how the relaxed problem defined in Section 2 can be adapted to take into account the additional terms in the objective function and the possible presence of the special roster. In particular, we show how to compute a lower bound on the number of technical intervals and an upper bound on the number of available days in the special roster. Since the objective function is of hierarchical type, these latter bounds are valid only if each bound on the terms of higher importance coincides with the corresponding optimal solution value.

As to the technical intervals, we proceed as follows. We first define the arc lengths in G in the ordinary way. Then, for each arc $(i,j) \in A_1$ such that the direct sequencing of j after i requires a technical interval, we add a sufficiently small positive value ϵ to length c^1_{ij}. Notice that here the value of LB does not necessarily correspond to an integer number of days. Let LB be the Lagrangian lower bound we obtain, and let $d := \lfloor LB/1440 \rfloor$ and $w := \lceil d/6 \rceil$ be the corresponding lower bounds on the number of days and weeks in a solution, respectively. If d is a multiple of 6, our lower bound t on the number of technical intervals is set to $(LB - (d \cdot 1440))/\epsilon$, which is the number of arcs associated with technical intervals in the AP solution. Otherwise, t is set to 0, since the addition of one day in the AP solution could avoid the use of the technical intervals, without increasing the number of weeks.

The possibility of introducing the special roster in the solution can also be taken into account, yielding a different lower bound LB' and an upper bound on the total number of available days in an optimal solution. Besides considering the problem defined by (2)-(13), we define a similar problem CRP$'$, where the use of the special roster is imposed. To this end, we introduce a "dummy" duty $n+1$, such that $s_{n+1} := 0$ and $p_{n+1} := 3 \cdot 1440$. This duty represents the last three complete days of the special roster, which have to be idle, the first one being an available day. Since any duty can immediately precede or follow this block, for $i = 1, \ldots, n$ we set $c^1_{i,n+1} := p_i + (1440 - f_i)$ and $c^1_{n+1,i} := p_{n+1} + s_i$, where $f_i = (s_i + p_i) \bmod 1440$ is the end time of duty i within a day. Also, no weekly rest can precede or follow this block, therefore for $i = 1, \ldots, n$ we set $c^2_{i,n+1} := c^3_{i,n+1} := c^2_{n+1,i} := c^3_{n+1,i} := +\infty$. Furthermore, since no weekly rest is required in the special roster, in problem CRP$'$ the total number of simple or double rest arcs in the cycles has to be at least equal

to the total length of the cycles, expressed in weeks, minus one. Constraint (9) is redefined accordingly. Problem LRP$'(\lambda_1, \lambda_2)$ is then defined from CRP$'$ and solved in the same way as LRP(λ_1, λ_2), yielding the Lagrangian lower bound LB', and the corresponding lower bounds d', w' and t' on the number of days, weeks and technical intervals, respectively, in an optimal solution where the special roster is imposed. The value $\tilde{LB} := \min\{LB, LB'\}$ is then a valid lower bound for the overall problem. Lower bounds on the number of days and weeks in an optimal solution are given by $\tilde{d} := \min\{d, d'\}$ and by $\tilde{w} := \min\{w, w'\}$, respectively. Also, a lower bound on the number of technical intervals is

$$\tilde{t} := \begin{cases} t & \text{if } w < w' \\ t' & \text{if } w > w' \\ \min\{t, t'\} & \text{if } w = w' \end{cases}, \tag{18}$$

and an upper bound on the number of available days is

$$\tilde{e} := \begin{cases} 0 & \text{if } w < w' \text{ or } (w = w' \text{ and } t < t') \\ (6 \cdot w' - d') + 1 & \text{otherwise} \end{cases} \tag{19}$$

(recall that one available day is included in the dummy duty $n + 1$).

Notice that it is easy to show that, with the numerical parameters defined in Sections 4.1, 4.2, and 4.3, LB' is in fact not less than LB.

4.7 Heuristic algorithm

Our heuristic algorithm follows the outline we gave in Section 3, with the following modifications related to the presence of technical intervals and available days in the objective function.

i) In the choice of the duty h to be sequenced after the current duty i in our heuristic procedure, the possible presence of a technical interval between i and h is also considered in the definition of score τ_h^1.

ii) Each time we consider the possibility of closing a roster of one week only, we check whether the roster can be considered as a special roster, and compute the corresponding number of available days. Among all rosters of one week in a complete solution, we consider as special roster the one having the largest number of available days.

5 Computational results

The lower and upper bounding procedures proposed in Sections 2 and 3, adapted to the FARO problem, have been implemented in FORTRAN. The resulting code was tested

		Lower Bound			BO			NA			TO		
Name	*n*	*weeks*	*t.i.*	*a.d.*	*weeks*	*t.i.*	*a.d.*	*weeks*	*t.i.*	*a.d.*	*weeks*	*t.i.*	*a.d.*
FARO164	164	48	0	1	48	0	0	48	5	0	49	2	0
FARO360	360	108	0	3	112	2	1	113	8	4	113	0	0
FARO525	525	164	0	1	165	0	2	168	5	3	168	2	0

Table 1: Results on the test instances of the FARO competition. Time limit of 90 minutes on a PC 486/33.

both on the real-world instances provided by Ferrovie dello Stato SpA within the FARO competition, and on "artificial" instances obtained by combining real-world instances.

Seven teams took part in the FARO competition. Table 1 illustrates the results obtained within the competition by the teams whose code was able to produce valid solutions for the three proposed test instances, namely:

BO Dipartimento di Elettronica, Informatica e Sistemistica, Università di Bologna (A. Caprara, M. Fischetti, P. Toth and D. Vigo);

NA Dipartimento di Informatica e Sistemistica, Università di Napoli (G. Bruno, G. Ghiani, G. Improta and M. Vento), see Bruno et al. (1995);

TO Dipartimento di Automatica e Informatica, Politecnico di Torino (C. Alfieri, P. Baracco, F. Della Croce, F. Rizzante, M. Sbodio and R. Tadei), see Tadei et al. (1995).

For each instance the table reports the number of weeks (*weeks*), technical intervals (*t.i.*) and available days (*a.d.*) of the heuristic solutions obtained by the three teams within a 90 minutes time limit on a PC 486/33 with 8 Mbytes of memory. We also report the lower bound obtained by our algorithm. The computing times required to obtain the reported solutions are 18, 76, and 25 minutes, respectively. For instances FARO164 and FARO525, we obtained tight lower and upper bound values. For FARO360, instead, the gap between the lower and upper bound values is larger. The main reason for this behavior is that the constraint on the paid time for this instance is much more binding than in the other cases. Typically, the Lagrangian lower bound $\tilde{w}$ on the number of weeks is much better than the trivial bounds (see Table 3). For instance FARO360, however, both $\tilde{w}$ and the trivial bound $\tilde{L}_1$ computed by taking into account only the maximum paid time of a roster, are equal to 108. For this reason, several rosters which are checked for feasibility in our heuristic turn out to be infeasible because they violate the paid time constraint. If this constraint is relaxed, we obtain for this instance a solution of value 108.

Name	n	*ext. rest*	*long*	*overnight*	*heavy*	$\overline{p}$	$\overline{w}$	$\overline{a}$
FARO021	21	5	3	7	2	575	446	575
FARO033	33	6	3	14	8	525	437	525
FARO069	69	13	2	18	2	507	411	507
FARO134	134	24	6	50	23	510	420	514
FARO386	386	100	22	171	66	560	429	576
FARO164	164	34	13	59	27	514	409	522
FARO360	360	92	52	136	49	587	456	610
FARO525	525	149	29	240	89	579	439	595

Table 2: Characteristics of the FARO instances.

Table 2 illustrates the characteristics of the real-world instances from the FARO competition, the first 5 distributed before the competition, and the last 3 actually used in the competition. For each instance the table reports:

Name	instance name;
n	number of duties;
ext. rest	number of duties with external rest;
long	number of long duties;
overnight	number of overnight duties;
heavy	number of heavy overnight duties;
$\overline{p}$	average duration of the duties (in minutes);
$\overline{w}$	average working time of the duties (in minutes);
$\overline{a}$	average paid time of the duties (in minutes).

Table 2 shows that a significant fraction of the duties have special attributes, for example more than one third of the duties are overnight.

In order to test the algorithm on larger and different test problems, additional instances have been obtained by merging all the pairs of distinct FARO instances. Table 3 illustrates the results obtained by the final version of our heuristic (the one described in this paper), containing several improvements with respect to the one used in the competition. For each instance we also report the values of the trivial lower bounds and the computing time to obtain the given solution. Time limits of 60 and 150 minutes have been imposed for the instances with $n \leq 500$ and $n > 500$, respectively. The computing times are expressed in PC Pentium 90 CPU seconds. Observe that the average computing times for obtaining the reported solution are about 15 and 49 minutes for the instances with $n \leq 500$ and $n > 500$, respectively. With a time limit of 300 minutes, the algorithm found for instances F164F525 and F134F360 solutions with 182 and 148 weeks, respectively.

The table clearly shows the effectiveness of the approach, for what concerns the lower

bound, the heuristic solution, and the computing time, also when applied to large size instances. For 26 out of 36 test problems the lower and upper bounds on the number of weeks coincide, i.e., the heuristic algorithm found solutions with the optimal number of crews. The average percentage gap between the heuristic solution value and the lower bound equals 0.8%. As previously mentioned, the Lagrangian lower bound is generally much better than the trivial bounds. Among these bounds, the one associated with the paid time gives the best value, which is on average 7.6% worse than the Lagrangian bound.

Acknowledgments

We are grateful to Ferrovie dello Stato SpA, in particular to Pier Luigi Guida, for having organized the FARO competition. We also thank the anonymous referees for their useful comments, and Stefano Martinotti and Alberto Dorati for their help in programming.

References

[1] AIRO, Ferrovie dello Stato SpA, "Metodi di Ottimizzazione per la Costruzione dei Turni - F.A.R.O.", Bando di Concorso (in Italian), July 1994.

[2] AIRO, Ferrovie dello Stato SpA, "Metodi di Ottimizzazione per la Costruzione dei Turni - F.A.R.O.", Verbale della Commissione Concorso FARO (in Italian), March 1995.

[3] N. Balakrishnan, R.T. Wong, "A Network Model for the Rotating Workforce Scheduling Problem", *Networks* 20 (1990) 25–42.

[4] L. Bianco, M. Bielli, A. Mingozzi, S. Ricciardelli, M. Spadoni, "A Heuristic Procedure for the Crew Rostering Problem", *European Journal of Operational Research* 58 (1992) 272–283.

[5] L. Bodin, B. Golden, A, Assad, M. Ball, "Routing and Scheduling of Vehicles and Crews: the State of the Art", *Computer and Operations Research* 10 (1983) 63–211.

[6] G. Bruno, G. Ghiani, G. Improta, M. Vento, "Un Algoritmo Greedy per la Soluzione del Rostering Problem: Il Caso delle FS", *Atti delle Giornate AIRO '95*, 1995.

[7] A. Caprara, M. Fischetti, P. Toth, "A Heuristic Algorithm for the Set Covering Problem", Technical Report, DEIS, University of Bologna, October 1995.

		Lower Bounds								Heuristic Solution			
		Trivial					*Lagrangian*						
Name	n	$\tilde{L}_1$	$\tilde{L}_2$	$\tilde{L}_3$	$\tilde{L}_4$	$\tilde{L}_5$	*weeks*	*t.i.*	*a.d.*	*weeks*	*t.i.*	*a.d.*	*time*
FARO021	21	6	4	3	4	5	7	0	3	7	0	3	8
FARO033	33	9	7	3	5	7	11	0	3	11	0	3	17
FARO069	69	18	9	2	10	14	19	0	2	19	0	0	650
FARO134	134	34	25	6	18	27	39	0	0	39	0	0	365
FARO164	164	43	30	13	25	32	48	0	1	48	0	0	106
FARO360	360	108	68	52	66	77	108	0	3	111	0	1	342
FARO386	386	110	86	22	72	77	118	0	2	118	0	0	443
FARO525	525	154	120	29	107	107	164	0	1	164	0	0	1185
F21F33	54	15	11	6	8	12	17	0	3	17	0	2	29
F21F69	90	24	13	5	13	18	25	0	3	26	0	2	180
F21F134	155	40	29	9	21	31	45	0	0	45	0	0	645
F21F164	185	48	33	16	28	36	54	0	0	54	0	0	103
F21F360	381	114	72	55	70	81	114	0	3	118	0	2	1805
F21F386	407	116	89	25	75	82	124	0	1	124	0	0	447
F21F525	546	160	124	32	110	112	170	0	2	170	0	0	1026
F33F69	102	26	16	5	14	20	29	0	3	29	0	1	533
F33F134	167	43	32	9	22	33	50	0	3	50	0	1	117
F33F164	197	51	37	16	29	38	58	0	1	58	0	0	149
F33F360	393	117	75	55	70	83	118	0	3	121	3	1	638
F33F386	419	118	93	25	76	84	128	0	3	128	0	1	1587
F33F525	558	162	127	32	111	114	174	0	2	174	0	0	1936
F69F134	203	51	34	8	27	40	57	0	0	58	0	0	1135
F69F164	233	60	39	15	34	45	66	0	2	66	0	0	327
F69F360	429	125	77	54	75	90	126	0	3	130	1	0	1892
F69F386	455	127	95	24	81	90	136	0	2	136	0	0	7462
F69F525	594	171	129	31	116	120	182	0	1	183^{*}	0	2	1848
F134F164	298	76	55	19	42	58	87	0	1	87	0	0	595
F134F360	494	142	93	58	83	103	146	0	3	149^{+}	0	1	3102
F134F386	520	143	111	28	89	103	157	0	2	157	0	1	1396
F134F525	659	188	145	35	124	133	203	0	2	203	0	0	2107
F164F360	524	150	98	65	90	108	155	0	2	158	0	0	2134
F164F386	550	152	115	35	96	108	166	0	3	166	0	0	956
F164F525	689	196	150	42	131	138	211	0	1	211	0	0	7715
F360F386	746	217	154	74	138	153	225	0	2	228	0	0	4434
F360F525	885	261	188	81	173	183	271	0	3	274	0	0	6898
F386F525	911	263	206	51	178	184	282	0	3	282	0	0	3806

Table 3: Results on the instances of our test-bed. Times expressed in PC Pentium 90 CPU seconds. (* solution with 182 weeks found in 11317 seconds, $^{+}$ solution with 148 weeks found in 16160 seconds).

[8] G. Carpaneto, P. Toth, "Primal-Dual Algorithms for the Assignment Problem", *Discrete Applied Mathematics* 18 (1987) 137–153.

[9] P. Carraresi, G. Gallo, "A Multilevel Bottleneck Assignment Approach to the Bus Drivers' Rostering Problem", *European Journal of Operational Research* 16 (1984) 163–173.

[10] M. Gamache, F. Soumis, "A Method for Optimally Solving the Rostering Problem", Les Cahiers du GERAD G-93-40, Montréal, 1993.

[11] M. Gamache, F. Soumis, G. Marquis, J. Desrosiers, "A Column Generation Approach for Large Scale Aircrew Rostering Problems", Cahiers du GERAD G-94-20, Montrèal, 1994.

[12] B. Hagberg, "An Assignment Approach to the Rostering Problem", in J.M. Rousseau (ed.), *Computer Scheduling of Public Transport 2*, North Holland, 1985.

[13] A.I.Z. Jarrah, J.T. Diamond, "The Crew Bidline Generation Problem", Technical Report, SABRE Decision Technologies, October 1995.

[14] J.K. Jachnik, "Attendance and Rostering Systems", in A. Wren (ed.) *Computer Scheduling of Public Transport*, North Holland, 1981.

[15] D.M. Ryan, "The Solution of Massive Generalized Set Partitioning Problems in Aircrew Rostering", *Journal of the Operational Research Society* 43 (1992) 459–467.

[16] R. Tadei, C. Alfieri, P. Baracco, F. Della Croce, F. Rizzante, M. Sbodio, "Un Algoritmo Genetico per il Problema di Rostering", *Atti delle Giornate AIRO '95*, 1995.

[17] J.M. Tien, A. Kamiyama, "On Manpower Scheduling Algorithms", *SIAM Review* 24 (1982) 275–287.