Algorithms for Railway Crew Management

Alberto Caprara [a], Matteo Fischetti [b], Paolo Toth [a],
Daniele Vigo [a] and Pier Luigi Guida [c]

[a] *DEIS, University of Bologna, Italy*

[b] *DMI, University of Udine, Italy*

[c] *Ferrovie dello Stato SpA, Italy*

Abstract

Crew management is concerned with building the work schedules of crews needed to cover a planned timetable. This is a well-known problem in Operations Research and has been historically associated with airlines and mass-transit companies. More recently, railway applications have also come on the scene, especially in Europe. In practice, the overall crew management problem is decomposed into two subproblems, called crew scheduling and crew rostering. In this paper, we give an outline of different ways of modeling the two subproblems and possible solution methods. Two main solution approaches are illustrated for real-world applications. In particular we discuss in some detail the solution techniques currently adopted at the Italian railway company, Ferrovie dello Stato SpA, for solving crew scheduling and rostering problems.

1 Introduction

Crew management is concerned with building the work schedules of crews needed to cover a planned timetable. This is part of more general activities within transit companies, broadly called *tactical planning*, which concern the medium-term use of the available resources. Crew management is a well-known problem in Operations Research and has been historically associated with airlines and mass-transit companies. More recently, railway applications have also come on the scene, especially in Europe where deregulation and privatization issues are increasingly pervading the rail industry and better productivity and efficiency are strongly required by the market and public ownership. This leads to railway companies being increasingly interested in using effective optimization techniques.

Preprint submitted to Elsevier Science *4 June 1997*

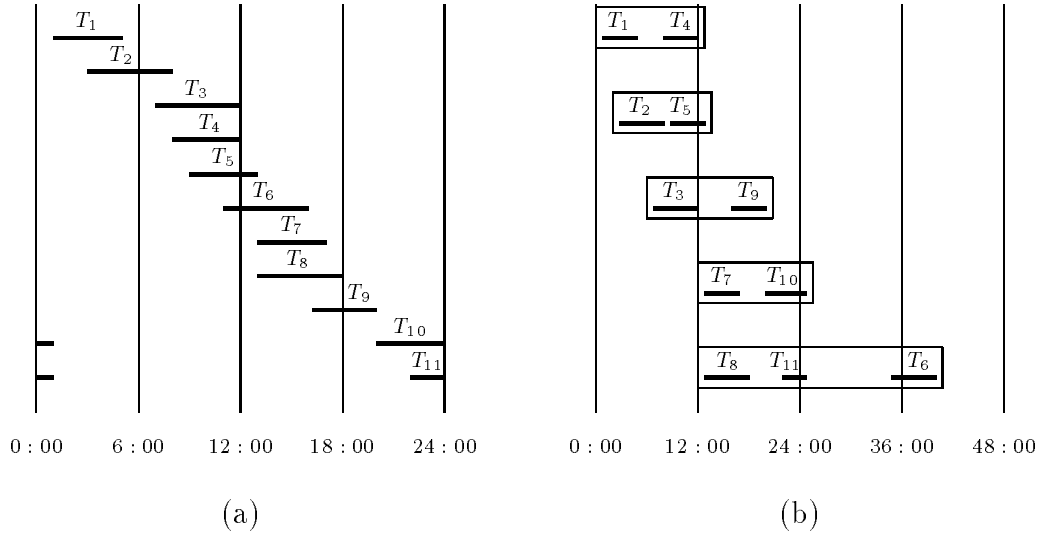

Fig. 1. (a) Trips to be covered every day. (b) Duties covering all the trips; each duty overlaps at most $L = 2$ consecutive days.

Several papers on crew (and vehicle) management appeared in the literature. We refer the interested reader to Wren [33], Bodin, Golden, Assad and Ball [10], Carraresi and Gallo [13], Daduna and Wren [16], Rousseau [28], Desrochers and Rousseau [17], Barnhart, Johnson, Nemhauser, Savelsbergh and Vance [4], Desrosiers, Dumas, Solomon and Soumis [18], and Wise [32].

In this paper we focus on crew management problems arising in railway applications, which can be outlined as follows. We are given a planned timetable for the *train services* (i.e., both the actual journeys with passengers or freight, and the transfers of empty trains or equipment between different stations) to be performed every day of a certain time period. Each train service has first been split into a sequence of *trips*, defined as segments of train journeys which must be serviced by the same crew without rest. Each trip is characterized by a departure time, a departure station, an arrival time, an arrival station, and possibly by additional attributes. Each daily occurrence of a trip has to be performed by one crew. In fact, each crew performs a *roster*, defined as a sequence of trips whose operational cost and feasibility depend on several rules laid down by union contracts and company regulations. The problem consists of finding a set of rosters covering every trip of the given time period, so as to satisfy all the operational constraints with minimum cost.

Figures 1(a) and 2 give a simple example of the problem. The trips to be covered every day are shown in Figure 1(a). An example of a roster covering these trips is illustrated in Figure 2. The roster consists of the cyclic trip sequence $T_3, T_9, \ldots, T_6, T_3, \ldots$, and spans 12 days. Each 6th day is left idle for crew rest. According to the roster, 12 crews are needed to perform each daily occurrence of the given trips. In fact, the first crew covers: on *calendar* day d, say, trips T_3 and T_9, on calendar day $d+1$ no trip, on calendar day $d+2$ trips T_2 and T_5, ..., on calendar day $d+11$ no trip, on calendar day $d+12$ again

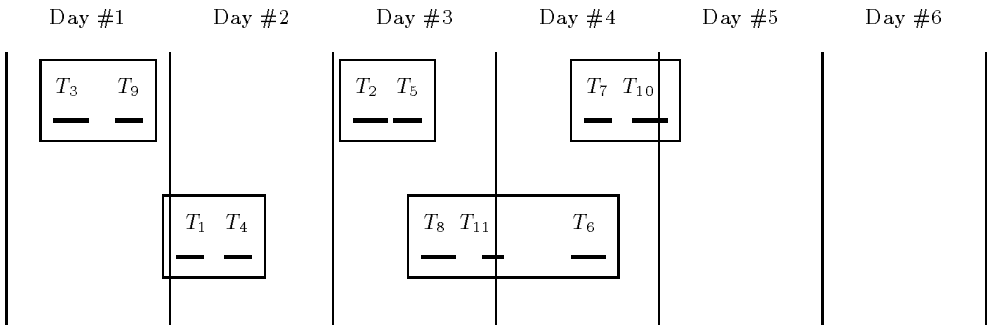

Fig. 2. A roster covering all the trips and requiring 12 crews. Boxes correspond to short-term subsequences (duties).

trips T_3 and T_9, and so on. On calendar day $d+1$, trips T_3 and T_9 are instead covered by the second crew, which performs no trips on day $d+2$, trips T_2 and T_5 on day $d+3$, and so on. Analogously, trips T_3 and T_9 on calendar day $d+2$ are covered by crew number 3, on calendar day $d+3$ by crew number 4, ..., on calendar day $d+11$ by crew number 12, and on calendar day $d+12$ by crew number 1 again.

Railway crew management represents a very complex and challenging problem due to both the size of the instances to be solved and the type and number of operational constraints. Typical figures at *Ferrovie dello Stato SpA*, the Italian railway company, are about 8,000 trains per day and a workforce of 25,000 drivers spread among several depots. The largest planning problems concern the inter-city and long-range passenger trains, and involve about 2,000 trains split into 5,000 trips per day.

In practice, the overall crew management problem is approached in two phases, according to the following scheme:

1. *Crew scheduling*: the short-term schedule of the crews is considered, and a convenient set of *duties* covering all the trips is constructed. Each duty represents a sequence of trips to be covered by a single crew within a given time period overlapping at most L consecutive days (e.g., $L = 2$). In the example in Figure 1(a), the trips are covered by means of the 5 duties reported in Figure 1(b).
2. *Crew rostering*: the duties selected in phase 1 are sequenced to obtain the final rosters. In this step, trips are no longer taken into account explicitly, but determine the *attributes* of the duties which are relevant for the roster feasibility and cost. In the example, the 5 duties in Figure 1(b) are sequenced to obtain the 12-day roster in Figure 2.

Decomposition is motivated by several reasons. First of all, each crew is located in a given *depot*, which represents the starting and ending point of its work segments. A natural constraint imposes that each crew must return to its home depot within L days, which leads to the concept of *duty* as a short-term

work segment starting and ending at the home depot and overlapping very few consecutive days. Secondly, constraints affecting the short-term work segments are different in nature from those related to the overall crew rosters. For example, in the Italian railway company the minimum time interval between two consecutive trips in a duty is a few minutes for changing trains, whereas the time interval between two consecutive duties is 18-22 hours for home rest. Finally, a global approach to the overall crew management problem is unlikely to be implemented, because of both its intrinsic difficulty and the planners' unwillingness to change their actual practice.

It is worth noting that crew rostering typically considers each depot separately, in that a roster cannot include duties associated with different crew home locations.

A main objective of crew management is the minimization of the global number of crews needed to perform all the daily occurrences of the trips in the given period. In some applications, the crew rostering phase plays a minor role, since the corresponding constraints are rather weak and the number of crews is easily determined from the solution of the crew scheduling phase. This happens, e.g., in urban mass transit applications, where crew rostering is aimed at balancing the workload among the crews as evenly as possible. As a result, the objective used in the crew scheduling phase mainly calls for the minimization of the number of working days corresponding to the duties.

In railway applications, instead, considerable savings can be obtained through a clever sequencing of the duties obtained in the first phase. Therefore, the objective of the crew scheduling phase has to take into account the characteristics of the duties selected and their implication in the subsequent rostering phase. This suggests the opportunity of feedback between the two phases, which allows for dynamically updating the crew scheduling costs.

This paper is organized as follows. Section 2 outlines different ways of modeling the problem and possible solution methods. Two main solution approaches are illustrated for real-world applications. To this aim, Sections 3 and 4 discuss in some detail the solution techniques currently adopted at the Italian railway company for solving crew scheduling and crew rostering subproblems, respectively.

2 Modeling and solution approaches

Both crew scheduling and crew rostering problems require finding min-cost *sequences* through a given set of *items*. Items correspond to trips for crew scheduling, and to duties for crew rostering. Sequences correspond to duties

for crew scheduling, and to rosters for crew rostering.

A natural formulation of both problems in terms of graphs associates a node with each item, and a directed arc with each possible item transition. More specifically, one can define a directed graph $G = (V, A)$ having one node $j \in V$ for each item, and an arc $(i, j) \in A$ if and only if item j can appear right after item i in a feasible sequence. With this representation, both problems can be formulated as finding a min-cost collection of *circuits* (or *paths*) of G covering each node once, as discussed in the sequel.

Consider first crew scheduling in the context of urban mass transit companies, where duty duration (*spread time*) is less than 24 hours. Here, a minimum duty start time b (e.g., 2 a.m.) is given. Accordingly, all departure/arrival times between 0 (midnight) and b are increased by 24 hours, and an arc $(i, j) \in A$ exists only if the arrival time of trip i is not greater than the departure time of trip j. This implies that G is acyclic. In contrast, in crew scheduling arising in railway applications graph G is not acyclic, and the departure and arrival times of the trips are intended modulo 24 hours. This allows an arc to connect a trip i to a trip j even if the arrival time of i is greater than the departure time of j, meaning that a crew performs trips i and j on different days. In both cases, crew scheduling calls for a min-cost collection of paths covering all the nodes once, each path satisfying a set of constraints related to the feasibility of the corresponding duty (maximum driving time, meal breaks, etc.). As already mentioned, a basic constraint for crew scheduling is that every duty must start and end at the crew home location (depot). It is then natural to introduce in G a dummy node d for each depot, along with the associated arcs (d, j) (respectively, (j, d)) for each node j associated with a trip which can be the first (resp., the last) trip in a duty assigned to depot d. This allows one to convert each path representing a duty into a circuit by connecting the terminal nodes of the path to the depot node representing the home location of the crew.

In crew rostering, the start and end times of the duties are intended modulo 24 hours, and the associated graph is not acyclic. No dummy depot nodes are needed, as all duties refer to the same depot. With an appropriate definition of the arc costs, the problem calls for a min-cost collection of circuits covering all the nodes once, each circuit satisfying a set of constraints related to the feasibility of the associated roster. This will be discussed in greater detail in Section 4.

There are two basic ways of modeling as an integer linear program the problem of covering the nodes of a directed graph through a suitable set of circuits. Let $\delta^+(v)$ and $\delta^-(v)$ represent the set of the arcs of G leaving and entering node $v \in V$, respectively.

The first model associates a binary variable x_{ij} with each arc $(i,j) \in A$, where $x_{ij} = 1$ if arc (i,j) is used in the optimal solution and $x_{ij} = 0$ otherwise. Let c_{ij} be the cost of each arc $(i,j) \in A$, and let $D \subset V$ denote the set of depot nodes (in crew rostering, $D = \emptyset$). The model reads:

$$\min \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{1}$$

subject to

$$\sum_{(i,j)\in\delta^+(v)} x_{ij} = \sum_{(i,j)\in\delta^-(v)} x_{ij} = 1, \qquad v \in V \setminus D \tag{2}$$

$$\sum_{(i,j)\in\delta^+(v)} x_{ij} = \sum_{(i,j)\in\delta^-(v)} x_{ij}, \qquad v \in D \tag{3}$$

$$\sum_{(i,j)\in P} x_{ij} \leq |P| - 1, \qquad P \in \mathcal{P} \tag{4}$$

$$x_{ij} \in \{0,1\}, \qquad (i,j) \in A \tag{5}$$

where family $\mathcal{P}$ contains the inclusion-minimal arc subsets P which cannot be part of any feasible solution ($|\mathcal{P}|$ may grow exponentially with $|V|$).

Constraints (2)–(3) impose that the same number of arcs enter and leave each node, and that each node not associated with a depot is covered exactly once. Constraints (4) forbid the choice of all the arcs in any infeasible arc subset P. Notice that $\mathcal{P}$ contains all the arc sequences which cannot be covered by a single crew because of operational constraints. In addition, $\mathcal{P}$ may contain subsets of arcs which cannot all be selected because of constraints related to the infeasibility of a group of circuits; these are typically called *crew base constraints*.

Model (1)–(5) has a number of drawbacks. First, it can only be applied when the cost of the solution can be expressed as the sum of the costs associated with the arcs. Hence it cannot be used when the cost of a circuit depends on the overall node sequence, or on the "type" of the crew, e.g., on the home location. Second, the linear programming relaxation of the model can be very weak when the operational constraints modeled by (4) are tight. This drawback can be partially overcome by introducing additional constraints taking into account explicitly some specific kinds of infeasibility. On the other hand, model (1)–(5) is particularly suitable for cases in which the most relevant constraints concern the direct transition of the nodes within the sequence, hence they can be effectively modeled through an appropriate definition of the arc set A and the arc costs c_{ij}.

A variant of the first model has a binary variable x_{ij}^k associated with each arc $(i,j) \in A$ and with each crew type k (typically k refers to depots), where $x_{ij}^k = 1$ means that a crew of type k covers nodes i and j in sequence. Let c_{ij}^k be the cost of $(i,j) \in A$ when performed by a crew of type k, where $c_{ij}^k = +\infty$ if (i,j) cannot be used by a crew of type k, and let K be the set of crew types. As before, D represents the (possibly empty) set of depot nodes. The model is:

$$\min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij}^k x_{ij}^k \tag{6}$$

subject to

$$\sum_{(i,j) \in \delta^+(v)} x_{ij}^k = \sum_{(i,j) \in \delta^-(v)} x_{ij}^k, \qquad v \in V,\ k \in K \tag{7}$$

$$\sum_{(i,j) \in P} x_{ij}^k \leq |P| - 1, \qquad P \in \mathcal{P}^k,\ k \in K \tag{8}$$

$$\sum_{k \in K} \sum_{(i,j) \in \delta^+(v)} x_{ij}^k = 1, \qquad v \in V \setminus D \tag{9}$$

$$x_{ij}^k \in \{0,1\}, \qquad (i,j) \in A,\ k \in K \tag{10}$$

where $\mathcal{P}^k$ is the family of all inclusion-minimal arc subsets P which cannot be part of any feasible solution for the crews of type k. With respect to the previous one, model (6)–(10) allows for arc costs depending on the crew type. Moreover, infeasibility constraints of type (8) can exploit the fact that the type of crew is given, which may lead to tighter linear programming relaxations. An obvious drawback is the increased size of the model, in terms of both the number of variables and constraints.

The second model has a possibly exponential number of binary variables, each associated with a feasible circuit of G. More specifically, let $\mathcal{C} = \{C_1, \ldots, C_n\}$ denote the collection of all the simple circuits of G corresponding to a feasible duty/roster for a crew, with $n = |\mathcal{C}|$. Each circuit C_j has an associated cost c_j, and covers the node set I_j. The binary variable y_j takes value 1 if C_j is part of the optimal solution, and 0 otherwise. We then have the following *set partitioning problem* with side constraints:

$$\min \sum_{j=1}^{n} c_j y_j \tag{11}$$

subject to

$$\sum_{j:v\in I_j} y_j = 1, \qquad v \in V \setminus D \tag{12}$$

$$\sum_{j\in S} y_j \leq |S| - 1, \qquad S \in \mathcal{S} \tag{13}$$

$$y_j \in \{0,1\}, \qquad j = 1,\ldots,n \tag{14}$$

where $\mathcal{S}$ denotes the family of all inclusion-minimal sets $S \subseteq \{1,\ldots,n\}$ with the property that no feasible solution contains all circuits C_j for $j \in S$. Constraints (12) impose that each node not associated with a depot is covered by exactly one circuit, whereas inequalities (13) model the crew base constraints.

A main advantage of the set partitioning model is that it allows for circuit costs depending on the whole sequence of arcs, and possibly on the crew type. Moreover, the feasibility constraints (13) need not take into account restrictions concerning the feasibility of a single circuit. As a result, they can often be replaced by a compact set of inequalities of the form $By \leq w$, modeling crew base constraints only. This produces a formulation whose linear programming relaxation is typically much tighter than in the previous models. Note however that the model often requires dealing with a very large number of variables. In some cases, the explicit generation of all feasible circuits is impractical, and one has to resort to a *column generation* approach, provided that an effective pricing procedure is available to find feasible circuits whose corresponding variable has a negative linear programming reduced cost.

In practice, the choice of the appropriate model and solution algorithm strongly depends on the particular structure of the problem in hand. According to our experience, the second model is particularly suitable for the cases in which feasible circuits cover a small number of nodes, and the constraints on the circuit feasibility are cumbersome and depend on the overall node sequence. This is the situation arising in railway crew scheduling, as described in Section 3. On the contrary, as already mentioned, the first model appears attractive for those cases where the main feasibility constraints concern the direct sequencing of two nodes, since they can be dealt with implicitly by an appropriate definition of the arc costs. This is the case of railway crew rostering, as described in Section 4.

3 Crew scheduling at the Italian railways

Due to the nature of the services to be carried out, in Italian railway applications a typical crew duty lasts no more than 24 hours and covers only a

few trips. Moreover, heavy operational constraints affect duty feasibility. This makes it practical to effect the explicit generation of all feasible duties, which are computed and stored in a preprocessing phase called *pairing generation*. In addition, operational rules allow a crew to be transported with no extra cost as a passenger on a trip, hence the overall solution can cover a trip more than once. In this situation, the set partitioning formulation (11)–(14) can profitably be replaced by its *set covering problem* relaxation obtained by replacing $=$ with $\geq$ in (12). As a result, only inclusion-maximal feasible duties, among those with the same cost, need be considered in the pairing generation. This considerably reduces the number of variables.

Even without side constraints (13), set covering problems arising in railway applications appear rather difficult, mainly because of their size. Indeed, the largest instances at the Italian railways involve up to $5,000$ trips and $1,000,000$ duties, i.e., they are 1-2 orders of magnitude larger than those arising in typical airline applications.

In 1994, the Italian railway company promoted the development of new techniques for an effective solution of very-large scale pure set covering instances. This resulted in a joint research project with the authors, which led to the heuristic approach described in the remaining part of this section. We refer the reader to Caprara, Fischetti and Toth [11] for more details.

The pure *Set Covering Problem* (SCP) can formally be defined as follows. Let $I_1, \ldots, I_n$ be the given collection of duties associated with the trip set $M = \{1, \ldots, m\}$. Each duty I_j has an associated cost $c_j > 0$. For notational convenience, we define $N = \{1, \ldots, n\}$ and $J_i = \{j \in N : i \in I_j\}$ for each trip $i \in M$. SCP calls for

$$v(\text{SCP}) = \min \sum_{j \in N} c_j y_j \tag{15}$$

subject to

$$\sum_{j \in J_i} y_j \geq 1, \qquad i \in M \tag{16}$$

$$y_j \in \{0, 1\}, \qquad j \in N \tag{17}$$

where $y_j = 1$ if duty j is selected in the optimal solution, $y_j = 0$ otherwise.

The exact SCP algorithms proposed in the literature can solve instances with up to few hundred trips and few thousand duties, see Beasley [5], Beasley and Jörnsten [8], and Balas and Carrera [2]. When larger instances are tackled, one has to resort to heuristic algorithms. Classical *greedy* algorithms are very fast

in practice, but typically do not provide high quality solutions, as reported in Balas and Ho [3] and Balas and Carrera [2]. The most effective heuristic approaches to SCP are those based on *Lagrangian relaxation* following the seminal work by Balas and Ho [3], and then the improvements by Beasley [6], Fisher and Kedia [20], Balas and Carrera [2], Ceria, Nobili and Sassano [15], and Wedelin [31]. Lorena and Lopes [27] propose an analogous approach based on *surrogate relaxation*. Recently, Beasley and Chu [7] and Jacobs and Brusco [25] proposed a genetic and a local search algorithm, respectively.

We next outline a heuristic method recently proposed by Caprara, Fischetti and Toth [11], which has been designed to attack very-large scale instances. The technique outperforms previously published methods: in 92 out of the 94 instances in the literature the method found, within short computing time, the optimal (or the best known) solution. Moreover, among the 22 instances for which the optimum is not known, in 6 cases the solution is better than any other solution found by previous techniques. The method is based on dual information associated with a Lagrangian relaxation of model (15)-(17). We refer to Fisher [19] for an introduction to Lagrangian optimization. For every vector $u \in R_+^m$ of Lagrangian multipliers associated with constraints (16), the Lagrangian subproblem reads:

$$L(u) = \min \left\{ \sum_{j \in N} c_j(u) y_j + \sum_{i \in M} u_i : \; y_j \in \{0,1\}, j \in N \right\} \tag{18}$$

where $c_j(u) = c_j - \sum_{i \in I_j} u_i$ is the *Lagrangian cost* associated with duty $j \in N$. Clearly, an optimal solution to (18) is given by $y_j(u) = 1$ if $c_j(u) < 0$, $y_j(u) = 0$ if $c_j(u) > 0$, and $y_j(u) \in \{0,1\}$ when $c_j(u) = 0$. The Lagrangian dual problem associated with (18) consists of finding a Lagrangian multiplier vector $u^* \in R_+^m$ which maximizes the lower bound $L(u)$. To solve this problem, a common approach uses the *subgradient vector* $s(u) \in R^m$ associated with a given u, defined by $s_i(u) = 1 - \sum_{j \in J_i} y_j(u)$ for $i \in M$. The approach generates a sequence $u^0, u^1, \ldots$ of nonnegative Lagrangian multiplier vectors, where u^0 is defined arbitrarily.

For near optimal Lagrangian multipliers u_i, the Lagrangian cost $c_j(u)$ gives reliable information on the overall utility of selecting duty j. Based on this property, we use Lagrangian (rather than original) costs to compute, for each $j \in N$, a *score* σ_j ranking the duties according to their likelihood to be selected in an optimal solution. These scores are given on input to a simple heuristic procedure, that finds a hopefully good SCP solution in a greedy way. Computational experience shows that almost equivalent near-optimal Lagrangian multipliers can produce SCP solutions of substantially different quality. In addition, no strict correlation exists between the lower bound value $L(u)$ and the quality of the SCP solution found. Therefore it is worthwhile applying the

heuristic procedure for several near-optimal Lagrangian multiplier vectors.

The approach consists of three main phases. The first one is referred to as the *subgradient phase*. It is aimed at quickly finding a near-optimal Lagrangian multiplier vector. To this end, an aggressive policy is used for the updating of the step-size and the reduction of the subgradient norm. The second one is the *heuristic phase*, in which a sequence of near-optimal Lagrangian vectors is determined. For each vector, the associated scores are given on input to a greedy heuristic procedure to possibly update the incumbent best SCP solution. In the third phase, called *fixing*, one selects a subset of duties having an estimated high probability of being in an optimal solution, and fixes to 1 the corresponding variables. In this way one obtains an SCP instance with a reduced number of duties (and trips), on which the three-phase procedure is iterated. After each application of the three-phase procedure, an effective *refining procedure* is used to produce improved solutions.

When very large instances are tackled, the computing time spent on the first two phases becomes very large. To overcome this difficulty, one can define a *core problem* containing a suitable set of duties, chosen among those having the lowest Lagrangian costs. The definition of the core problem is often very critical, since an optimal solution typically contains some duties that, although individually worse than others, must be selected in order to produce an overall good solution. Hence it is better not to "freeze" the core problem, and use a *variable pricing* scheme to update the core problem iteratively in a vein similar to that used for solving large scale linear programs. The use of pricing within Lagrangian optimization drastically reduces computing time, and is one of the main ingredients for the success of the overall scheme.

The Caprara-Fischetti-Toth algorithm, hereafter called CFT, was tested on the real-world instances provided by the Italian railway company within the competition FASTER, aimed at developing effective heuristics for very-large scale SCP instances. Table 1 reports the corresponding results. For each instance the table gives the instance name, the number of trips (m) and duties (n), the density $\sum_{j \in N} |I_j|/(m \cdot n)$, the value of the lower bound LB computed by the subgradient procedure, the value of the heuristic solution found by code CFT, and the best solution obtained by other methods. The reported solutions were obtained within time limits of 3,000 CPU seconds on a PC 486/33 for the first three instances, and 10,000 CPU seconds on a HP 9000 735/125 for the remaining instances.

The table shows that algorithm CFT is capable of providing near-optimal solutions within limited computing time even for very-large size instances. The average percentage gap between the lower bound and the heuristic solution value is 0.9%.

Table 1
Results on crew scheduling instances from Ferrovie dello Stato SpA.

Name	$m \times n$	Density	LB	CFT Sol.	Others' Sol.
FASTER507	$507 \times 63,009$	1.2%	173	174	174
FASTER516	$516 \times 47,311$	1.3%	182	182	182
FASTER582	$582 \times 55,515$	1.2%	210	211	211
FASTER2536	$2,536 \times 1,081,841$	0.4%	685	691	692
FASTER2586	$2,586 \times 920,683$	0.4%	937	947	951
FASTER4284	$4,284 \times 1,092,610$	0.2%	1051	1065	1070
FASTER4872	$4,872 \times 968,672$	0.2%	1509	1534	1534

4 Crew rostering at the Italian railways

Most of the published works on the crew rostering problem refer to urban mass-transit systems, where the minimum number of crews required to perform the duties can easily be determined, and the objective is to evenly distribute the workload among the crews: see Jachnik [24], Bodin, Gloden, Assad and Ball [10], Carraresi and Gallo [14], Hagberg [23], and Bianco, Bielli, Mingozzi, Ricciardelli and Spadoni [9]. Set partitioning approaches for airline crew rostering are described in Ryan [29], Gamache and Soumis [21], Gamache, Soumis, Marquis and Desrosiers [22], and Jarrah and Diamond [26]. Finally, related cyclic scheduling problems are dealt with in Tien and Kamiyama [30], and Balakrishnan and Wong [1].

We next give a description of the real-world crew rostering problem arising at the Italian railways. 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, and an *end time*, f_i (with $0 \leq s_i < 1440$ and $0 \leq f_i < 1440$, where 0 corresponds to midnight). Let p_i denote the *spread time* of duty i, i.e., the total time between the start and the end of the duty. Moreover, each duty i has an associated *working time*, w_i, which is the time actually spent working during the duty, and a *paid time*, a_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). Each duty can have additional characteristics, which are explicitly given on input:

- *duty with external rest*, if it includes a long 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 it requires some working between midnight and 5 am;
- *heavy overnight duty*, if it is an overnight duty without external rest, and

requires more than 1 hour and 30 minutes' work between midnight and 5 am.

A roster contains a subset of duties and spans a cyclic sequence of groups of 6 consecutive days, conventionally called *weeks*. Hence the number of days in a roster is an integer multiple of 6. The length of a roster is typically 30 days (5 weeks) and does not exceed 60 days (10 weeks), although these requirements are not explicitly imposed as constraints.

The crew rostering problem consists of finding a feasible set of rosters covering all the duties and spanning a minimum number of weeks. As already discussed in the introduction, the global number of crews required every day to cover all the duties is equal to 6 times the total number of weeks in the solution. Thus, the minimization of the number of weeks implies the minimization of the global number of crews required.

4.1 Operational constraints

For short, we call *complete day* a time interval of 24 hours (i.e., 1440 minutes) starting at midnight. Moreover, a complete day is called *free* if no duty or part of a duty is performed during that day.

Each week can include at most the following number of duties having particular characteristics: (i) 2 duties with external rest; (ii) 1 long duty; (iii) 2 overnight duties. Furthermore, each week must be separated from the next one in the roster by a continuous (weekly) rest, which always spans the complete sixth day of the week. There are two types of rests, conventionally called *simple* and *double* rests. Simple rests must be at least 48 hours long, whereas double 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 following one.

For each roster, the number of double rests must be at least 40% of the total number of rests, and the average rest time must be at least 58 hours. Moreover, for each (cyclic) group of 30 consecutive days within a roster, no more than 7 duties with external rest can be included, and 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.

4.2 Sequencing rules

Two consecutive duties of a roster, say i and j, can be sequenced either *directly* in the same week, or with a simple or double rest between them.

The break between the end of a duty and the start of the subsequent duty within a week lasts at least 18 hours. If both duties are overnight and at most one of them is a heavy overnight duty, the minimum break lasts 22 hours, while if both are heavy overnight duties the break must span at least one complete day. Moreover, after two consecutive overnight duties in a week whose intermediate break does not span a complete day, the break before the start of any other duty in the same week must last at least 22 hours.

When a simple 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. Finally, if the first duty in a week following a double rest starts before 6 am, then the rest must span at least three complete days.

4.3 Lower bounds

Simple lower bounds can easily be obtained by considering each of the operational constraints imposing a limit either on the total number of duties with a given characteristic, or on the total working and paid time in a week and in a (cyclic) group of 30 consecutive days in a roster, respectively.

A more sophisticated relaxation is proposed by Caprara, Fischetti, Toth and Vigo [12] in order to take into account all the rules for sequencing two consecutive duties within a roster. The relaxation also imposes that the total number of rests is equal to the total number of weeks making up the rosters, and that the total number of double rests is at least 40% of the total number of rests. We next give a graph theory description of the resulting relaxation, hereafter called RP.

We are given a complete directed multigraph $G = (V, A)$, where each node in $V = \{1, \ldots, n\}$ is associated with a duty. The arcs represent the consecutive sequencing of duty pairs within a roster. As previously described, two duties can be sequenced in three different ways, namely directly, or with a simple or double rest between them. Accordingly, arc set A contains arcs of three different types and can be partitioned into three subsets, A_1, A_2 and A_3. For each pair of nodes $i, j \in V$ we have an arc $(i, j) \in A_1$, whose cost 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. Similarly, we have an arc $(i, j) \in A_2$ (resp., $(i, j) \in A_3$) whose cost c^2_{ij} (resp., c^3_{ij}) is

the minimum time between the start of duty i and the start of duty j when a simple (resp., double) rest is imposed between them. G has no loops, therefore we let $c^1_{ii} = c^2_{ii} = c^3_{ii} = +\infty$ for each $i \in V$. In the sequel, arcs belonging to A_1 are also called *direct arcs*, while arcs belonging to A_2 (resp., A_3) are called *simple-rest arcs* (resp., *double-rest arcs*). Matrices c^1, c^2 and c^3 can easily be computed from the input data, according to the sequencing rules. Notice that, by definition of the arc costs, for a given pair $i, j \in V$ the values $c^1_{ij}, c^2_{ij}, c^3_{ij}$ differ by integer multiples of 1440.

Each circuit of G corresponds to a (possibly infeasible) roster, the cost of the circuit being the time required to perform the corresponding duties. Problem RP then calls for the determination of a minimum-cost set of disjoint circuits of G satisfying the following constraints:

- each node of G is covered by exactly one circuit;
- the total number of simple- or double-rest arcs in the circuits has to be at least the total cost of the circuits, expressed in weeks;
- the total number of double-rest arcs in the circuits has to be at least 0.4 times the total number of simple- or double-rest arcs.

Problem RP can be formulated as the following integer linear program. For each arc $(i, j) \in A_l$, $l = 1, 2, 3$, one introduces a binary variable x^l_{ij}, equal to 1 if arc $(i, j) \in A_l$ is in the optimal solution, and 0 otherwise. Moreover, an integer variable r represents the minimum number of simple- or double-rest arcs in the solution, and an integer variable z represents the minimum number of double-rest arcs in the solution. Let $\alpha = 6 \cdot 1440$ be the number of minutes in a week. The model reads:

$$v(\mathrm{RP}) = \min \sum_{i=1}^{n} \sum_{j=1}^{n} (c^1_{ij} x^1_{ij} + c^2_{ij} x^2_{ij} + c^3_{ij} x^3_{ij}) \tag{19}$$

subject to

$$\sum_{i=1}^{n} (x^1_{ij} + x^2_{ij} + x^3_{ij}) = 1, \qquad j = 1, \ldots, n \tag{20}$$

$$\sum_{j=1}^{n} (x^1_{ij} + x^2_{ij} + x^3_{ij}) = 1, \qquad i = 1, \ldots, n \tag{21}$$

$$r \geq \frac{1}{\alpha} \sum_{i=1}^{n} \sum_{j=1}^{n} (c^1_{ij} x^1_{ij} + c^2_{ij} x^2_{ij} + c^3_{ij} x^3_{ij}) \tag{22}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} (x^2_{ij} + x^3_{ij}) \geq r \tag{23}$$

$$z \geq 0.4\, r \tag{24}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij}^3 \geq z \tag{25}$$

$$x_{ij}^1, x_{ij}^2, x_{ij}^3 \in \{0,1\}, \qquad i,j = 1,\ldots,n \tag{26}$$

$$r, z \geq 0 \text{ integer.} \tag{27}$$

Constraints (20) and (21) impose that each node has exactly one entering and one leaving arc, respectively. Constraints (22) and (23) ensure that the total number of simple- or double-rest arcs is at least the total cost of the solution, expressed in weeks. Similarly, constraints (24) and (25) ensure that the total number of double-rest arcs is at least 0.4 times the total number of simple- and double-rest arcs.

Note that $v(RP)$ is expressed in minutes, but due to the structure of the arc costs, it always corresponds to an integer number of days, which represents a lower bound on the total number of days required to cover all the duties. Hence r is a lower bound on the number of weeks in an optimal solution.

A Lagrangian lower bound on v(RP) can be obtained as follows. One first relaxes in a Lagrangian way constraints (23) and (25), with nonnegative Lagrangian multipliers λ_1 and λ_2, respectively, obtaining the objective function

$$\min \lambda_1 r + \lambda_2 z + \sum_{i=1}^{n}\sum_{j=1}^{n}(\bar{c}_{ij}^1 x_{ij}^1 + \bar{c}_{ij}^2 x_{ij}^2 + \bar{c}_{ij}^3 x_{ij}^3), \tag{28}$$

where $\bar{c}_{ij}^1 = c_{ij}^1$, $\bar{c}_{ij}^2 = c_{ij}^2 - \lambda_1$, and $\bar{c}_{ij}^3 = c_{ij}^3 - \lambda_1 - \lambda_2$ are the *Lagrangian costs* for the x-variables. Furthermore, one replaces constraint (22) with its relaxation:

$$r \geq \frac{1}{\alpha}\left(\lambda_1 r + \lambda_2 z + \sum_{i=1}^{n}\sum_{j=1}^{n}(\bar{c}_{ij}^1 x_{ij}^1 + \bar{c}_{ij}^2 x_{ij}^2 + \bar{c}_{ij}^3 x_{ij}^3)\right). \tag{29}$$

Let LRP(λ_1, λ_2) denote problem (28), (20), (21), (29), (24), (26) and (27), and let v(LRP(λ_1, λ_2)) be its optimal solution value. Given multipliers $\lambda_1, \lambda_2 \geq 0$, v(LRP(λ_1, λ_2)) is a valid lower bound on v(RP) which can be computed by decomposing the problem into three subproblems associated with variables x, r, and z, respectively. This requires:

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

ii) determining the minimum r such that $\alpha r \geq \lambda_1 r + \lambda_2 \lceil 0.4r \rceil + v(\mathrm{AP})$, $r \geq 0$ integer;
iii) defining $z = \lceil 0.4r \rceil$.

Thus, the overall time complexity for solving $\mathrm{LRP}(\lambda_1, \lambda_2)$ is $O(n^3)$. Computational experience has shown that a tight lower bound can be computed as $v^* = \max\{v(\mathrm{LRP}(1440, 1440)),\ v(\mathrm{LRP}(1440, 0))\}$, by solving only two APs.

4.4 The heuristic algorithm

In this subsection we outline the constructive heuristic of [12], which extensively uses the information obtained from the solution of the relaxed problem defined in the previous subsection. The heuristic constructs one feasible 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 describe the procedure we use to build each single roster, as it applies to the construction of the first roster.

One first computes the lower bound v^* described in the previous subsection, and then starts building the roster by selecting its "initial duty" i_0 which will be performed at the beginning of a week, i.e., preceded by a rest. (The term "initial" is conventional as rosters are cyclic.) Once the initial duty has been selected, a sequence of iterations is performed where:

a) the best duty j to be sequenced after the current duty i is chosen, according to an appropriate *score* taking into account the lower bound increase due to the choice of arc $(i, j)^l$ and the characteristics of duty j;
b) the Lagrangian lower bound is parametrically updated, in $O(n^2)$ time;
c) the possibility of "closing" the roster is considered, possibly updating the best roster found.

The procedure is iterated until no better roster can be constructed, stopping anyway if the current roster spans more than 10 weeks.

When a complete solution to the problem is found, one can try to improve it by applying a *refining procedure*, which removes the last rosters constructed (which are typically worse than the others) from the solution, and re-applies the heuristic algorithm to the corresponding duties. To this end, some parameters of the roster construction procedure are either changed with a random perturbation or tuned so as to take into account the constraints that made the construction of the last rosters difficult.

Table 2
Results on crew rostering instances from Ferrovie dello Stato SpA.

		Lower Bounds		Heuristic Solution	
Name	n	*Simple*	*Lagrangian*	weeks	time
FARO021	21	6	7	7	8
FARO033	33	9	11	11	17
FARO069	69	18	19	19	650
FARO134	134	34	39	39	365
FARO164	164	43	48	48	106
FARO360	360	108	108	111	342
FARO386	386	110	118	118	443
FARO525	525	154	164	164	1185

4.5 Computational results

The previously described lower and upper bounding procedures were tested on real-world instances provided by the Italian railway company within the competition FARO, aimed at developing effective heuristics for crew rostering. The results obtained are illustrated in Table 2. For each instance we report the instance name, the number of duties, the best simple lower bound (as introduced at the beginning of subsection 4.3), the Lagrangian lower bound, the heuristic solution value, and the corresponding computing time, expressed in PC Pentium 90 CPU seconds.

The table clearly shows the effectiveness of the approach, since 6 out of 7 instances have been solved to proven optimality within no more than 20 minutes.

References

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

[2] E. Balas and M.C. Carrera, "A Dynamic Subgradient-Based Branch-and-Bound Procedure for Set Covering", *Operations Research* 44 (1996) 875–890.

[3] E. Balas and A. Ho, "Set Covering Algorithms Using Cutting Planes, Heuristics and Subgradient Optimization: A Computational Study", *Mathematical Programming Study* 12 (1980) 37–60.

[4] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh and P.H. Vance, "Branch–and–Price: Column Generation for Solving Huge Integer Programs", in J.R. Birge and K.G. Murty (eds.), *Mathematical Programming: State of the Art 1994*, The University of Michigan, 1994, 186–207.

[5] J.E. Beasley, "An Algorithm for Set Covering Problems", *European Journal of Operational Research* 31 (1987) 85–93.

[6] J.E. Beasley, "A Lagrangian Heuristic for Set Covering Problems", *Naval Research Logistics* 37 (1990) 151–164.

[7] J.E. Beasley and P.C. Chu, "A Genetic Algorithm for the Set Covering Problem", *European Journal of Operational Research* 94 (1996) 392–404.

[8] J.E. Beasley and K. Jörnsten, "Enhancing an Algorithm for Set Covering Problems", *European Journal of Operational Research* 58 (1992) 293–300.

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

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

[11] A. Caprara, M. Fischetti and P. Toth, "A Heuristic Method for the Set Covering Problem", Technical Report OR-95-8, DEIS University of Bologna, 1995, extended abstract published in W.H. Cunningham, S.T. McCormick and M. Queyranne (eds.) *Proceedings of the Fifth IPCO Conference*, Lecture Notes in Computer Science 1084, Springer, 1995, 72–84.

[12] A. Caprara, M. Fischetti, P. Toth and D. Vigo, "Modeling and Solving the Crew Rostering Problem", Technical Report OR-95-6, DEIS University of Bologna, 1995, to appear in *Operations Research*.

[13] P. Carraresi and G. Gallo, "Network Models for Vehicle and Crew Scheduling", *European Journal of Operational Research* 16 (1984) 139–151.

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

[15] S. Ceria, P. Nobili and A. Sassano, "A Lagrangian-Based Heuristic for Large-Scale Set Covering Problems", Technical Report R.406, IASI-CNR, Rome, 1995, to appear in *Mathematical Programming*.

[16] J.R. Daduna and A. Wren (eds.), *Computer-Aided Transit Scheduling*, Lecture Notes in Economic and Mathematical Systems 308, Springer Verlag, 1988.

[17] M. Desrochers and J.-M. Rousseau (eds.), *Computer-Aided Transit Scheduling*, Lecture Notes in Economic and Mathematical Systems 386, Springer Verlag, 1992.

[18] J. Desrosiers, Y. Dumas, M.M. Solomon and F. Soumis, "Time Constrained Routing and Scheduling", in M.O. Ball et al. (ed.s), *Handbooks in OR & MS*, Vol. 8, Elsevier Science, 1995, 35–139.

[19] M.L. Fisher, "The Lagrangian Relaxation Method for Solving Integer Programming Problems", *Management Science* 27 (1981) 1–18.

[20] M.L. Fisher and P. Kedia, "Optimal Solutions of Set Covering/Partitioning Problems Using Dual Heuristics", *Management Science* 36 (1990) 674–688.

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

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

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

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

[25] L.W. Jacobs and M.J. Brusco, "A Local Search Heuristic for Large Set-Covering Problems", *Naval Research Logistics* 52 (1995) 1129–1140.

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

[27] L.A.N. Lorena and F.B. Lopes, "A Surrogate Heuristic for Set Covering Problems", *European Journal of Operational Research* 79 (1994) 138–150.

[28] J.-M. Rousseau (ed.), *Computer Scheduling of Public Transport 2*, North Holland, 1985.

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

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

[31] D. Wedelin, "An Algorithm for Large Scale 0-1 Integer Programming with Application to Airline Crew Scheduling", *Annals of Operational Research* 57 (1995) 283–301.

[32] T.H. Wise, "Column Generation and Polyhedral Combinatorics for Airline Crew Scheduling", Ph.D. thesis, Cornell University, 1995.

[33] A. Wren (ed.), *Computer Scheduling of Public Transport*, North Holland, 1981.