Large-Scale Rostering in the Railway Domain

Tiago Maduro Dias
SISCOG - Sistemas Cognitivos Lda, Campo Grande 378 - 3º, 1700-097 Lisboa, Portugal
Instituto Superior Técnico, Av. Rovisco Pais 1, 1049-001 Lisboa, Portugal
tmdias@ieee.org

Abstract. The Rostering Problem deals with the assignment of work to resources over a period of time, subject to several, in some cases complex, labour and operational constraints. This problem, more specifically its variant Crew Rostering Problem (CRP), is of particular significance for mass transit companies, such as commercial airlines and railways. In this work, a general solving method for large scale instances of the CRP is proposed. The method is tested on the real planning scenario of an European railway company. The obtained results improve significantly on those achieved by specialized human planners, while being more generic than the solutions currently available in the literature.

Keywords: artificial intelligence, combinatorial optimisation, crew rostering problem, mixed-integer linear programming, operations research.

1 Introduction

The assignment of work to one or more resources is a complex task, particularly in domains like those of mass transit. In these realities, tasks are grouped to form more manageable duties, which are then assigned to resources, in the so-called Rostering Problem. In order to efficiently use the resource’s effort, these duties are normally very diverse in regard to their characteristics (e.g., duration, start/end times). This, together with their dimension and complex labour and operational constraints, make the Rostering Problems in mass transit companies, like railways and commercial airlines, especially challenging.

In this work, a generic method for solving very-large scale rostering problems is proposed. The method is tested with the real planning scenario of an European railway company, in this case corresponding to a Crew Rostering Problem (CRP), i.e., a rostering problem applied to personnel resources. The idea of the method being easily adaptable to several realities is introduced and, although this work focuses on the CRP, it should be noted that most of the considerations are generic and could be adapted to other types of resources, like rolling stock in railways.

This document is divided as follows: Section 2 describes the CRP and its particularities in more detail. The main efforts found in the literature to address this problem and related variants are presented in Section 3. The proposed solution is described in Section 4 and the obtained results in Section 5. Some final conclusions and ideas for future research on the proposed method can be found in Section 6.
2 The Crew Rostering Problem

The CRP can be formulated in a relatively straightforward way: given a domain of planning elements, i.e., units of interest for the planning problem, such as a duty or a day off, find their optimal distribution over one or more rosters, during a certain period of time. There are, however, several constraints that make the problem difficult to solve:

- the produced rosters must be valid with respect to the labour rules agreed with the unions;
- they must cater to the resources, particularly personnel resource preferences and satisfaction;
- the resulting solutions must meet the operational goals of the company.

More often than not, the operational goals conflict with the other constraints, as do the constraints compete with each other. Furthermore, the problem is characterized by an exponential growth of the search space and the existence of many feasible solutions, a number of which with good quality (corresponding to several local maxima). All of this means that, finding optimal or near-optimal solutions becomes a difficult combinatorial problem, which Chuin Lau shows to be NP-Hard [CL96].

There are two main variants for the CRP: production of cyclic and non-cyclic (individual) rosters. A cyclic roster, also known as rota [SN04], is used in complex domains but with a great operational regularity, like the railways. In this case, the timetable that will be used for several months (e.g., the whole summer) is known in advance, so it is possible to prepare what can be seen as a baseline of the overall plan of operations, called the long-term plan, which will be progressively refined as the time of operation approaches. The cyclic roster implicitly specifies the work of several resources during the timetable period in an efficient way. It is composed of several lines of a predefined length (normally a week), to which duties and other day types (e.g., days off or reserve days) are assigned in such a way that, if planned sequentially, the lines would form a valid individual roster. Once a cyclic roster is built, resources are assigned to each line and perform the corresponding work, continuing to the next line and so on. The resource that performs the last line of the cyclic roster goes back to the first. It is this cyclic nature of the roster that gives it its name. This process repeats itself until the period in which the timetable is effective ends and a new cyclic roster comes into play based on a new timetable. Note that, since the cyclic rosters are used to base the work of several resources over an entire timetable period, their quality has a significant impact on the railway operation.

In order to better understand this concept, Figure 1 presents a hypothetical example of a cyclic roster, where duties are shown as lines and the remaining days are days off ("O") and reserve days ("R"), days where the resource is on prevention and will normally be assigned work closer to the operation. There are 3 individual rosters that may be derived from the presented example. For instance, the worker starting on the second line would perform the following work during the first three weeks: 1 1 2 O 1 2 R R 2 R 1 O O 2 2 O 1 2 2 O O …
Fig. 1. Cyclic roster example.

3 Related Work

The literature is yet to reveal a clear trend for what is considered a suitable generic approach to solve the CRP (or the CCRP). Several methods have been suggested, mostly employing Operations Research (OR) techniques, known for their competence in solving difficult combinatorial problems.

Kohl and Karisch describe in [KK04] the generic solution used in the commercial application developed by Jeppesen Sanderson for the planning of non-cyclic rosters. This solution employs the principles of Column Generation to solve the problem modelled as a Set Partitioning Problem. Similar methods (also based on Column Generation) are studied in [GSV+98], [GSMD99] and [GS93]. Sellmann and others suggest a method in [SZSF02] and [SZSF00] that employs Constraint Satisfaction to support the generation of columns in highly constrained scenarios. In [CG04], Cappanera and Gallo employ a multicommodity-flow model to the problem, with interesting success for small to medium-sized problems.

Unlike what happens in the aviation, for invididual rosters, it is difficult to find quality advancements in the cyclic rostering problem. The first relevant proposal only appears in 1995, in [CTVF98], with the work that won the FARO competition organized by the Italian railways. The generic approach employs a heuristic search that uses information taken from the solution of a mathematical model of the problem, which finds the optimal path between a connected graph with all the duties in the problem (note that the correct problem statement would require finding n individual paths between the duties in order to fill the n rosters in the problem). This approach is further refined in [CFL+98], employing Constraint Satisfaction. An interesting development appears in [CMT01], where an attempt is made to combine the generation of duties and production of rosters.

In [SN04] and [HHAK06], an alternative approach is introduced, where the planning of rosters is divided into several stages: (1) days-off planning; (2) pattern assignment; and (3) duty assignment. The method proposed in this work may be seen as a generalization and extension of these methods.
4 Approach

The challenge of this work is to find a solving method that can address, in an effective way, large-scale instances of the CRP, while also being adaptable to smaller ones. Furthermore, it is important that the method is generic enough to be applicable to a wide range of formulations of the problem, from different realities, not just the proposed test instance.

The test instance is presented in Section 4.1. The subsequent sections describe the process (Section 4.2) and the process phases (Sections 4.3 and 4.4). The way the problem is modelled in the constructive phase as a MIP is presented in Section 4.5.

4.1 Test Instance

To test the method proposed in this work, a problem is taken from the real planning scenario of a European railway company. The problem consists of solving a large instance of a Cyclic Crew Rostering Problem (CCRP), where 1459 duties are to be planned over 304 weeks distributed in 23 rosters, several of which with a preference over a specific type of duty.

To justify the classification of large instance, note that, if all duty combinations were valid, there would be, on average, 107 duties that could be planned on each roster day. This translates into an approximate state space of $10^{2128}$. Even if this number is reduced by half, considering all constraints, it would still be a number far larger than, for example, the estimated state space of chess ($10^{43}$) or the estimated number of particles in the universe ($10^{87}$)!

As mentioned, for a solution to be considered valid, several constraints must be validated. Another aspect that makes this type of problems difficult to solve is the existence of many constraints that are global in the sense that they can only be validated over completely instantiated rosters (which practically rules out the use of pure constructive strategies). For instance:

- maximum number of duties over 31 days;
- work capacity of consecutive weeks; and
- work capacity per roster.

All in all, for the complete problem, 9516 feasibility constraints have to be validated and approximately $36^{152}$ alternative day off pattern combinations exist.

Adding to these characteristics, it must be taken into account that the goal is not to merely build optimized rosters with regards to maximizing the performed work. Because the resulting base plan influences a large period of time, during which a number of unforeseeable events may occur, special care must be taken to make the plan robust. This is achieved by balancing the optimization with buffers over certain aspects of the plan that are particularly prone to adjustments nearer to the time of operation (the so-called short-term planning or real-time dispatching). Regarding this topic, an important

---

1 From this point on, the term roster refers to cyclic roster, unless explicitly stated otherwise. Note, however, that the problem formulations are similar enough for most of the considerations to also be applicable to individual rosters.
additional goal is to leave consecutive duties at least 13 hours apart, so that it is possible for a resource to come in early or perform an additional task at the end of a duty without violating the minimum amount of rest time of 11 hours. On the other hand, because resources must be kept (sufficiently) happy, they should, for instance, be allowed to go home at a reasonable hour before a day off as well as not be assigned early duties after one. As can be easily noted, these goals all compete against each other. For example, the robustness driven minimum of 13 hours gap between duties leads to solutions where the duties are spaced out, while meeting the aforementioned protection times around days off means leaving them closer together (so they can be further away from the days off). These are examples of the competing goals that were mentioned above.

All in all, an effective solution method must achieve a good balance between having a global overview of the problem, to handle the global constraints and competing goals, while keeping some ability of optimizing the detail, in a very (very) large state space.

4.2 Process

Linear Programming (LP) is popular for its ability to find optimal compromises between competing goals and global constraints, both commonly found in planning problems, like the CRP. On the other hand, this technique is somewhat sensitive to the exponential growth of the state space and can only represent linear constraints and cost functions. The first aspect, in particular, implies that some sort of relaxation over the state space would be needed, possibly with implications in the quality of the obtained solution. To cope with this issue, it is suggested that a Local Search (LS), as studied in Artificial Intelligence [Pea84], is run from the resulting relaxed solution, leading to the process shown in Figure 2. The underlying idea is to combine the LS’s ability to handle the full complexity of the problem (including non-linear constraints and cost function) with LP’s potential of at least converging on a vicinity of a global maximum. Note that, given the size of the state space, it would be very difficult for a LS to systematically avoid the several local maxima of the problem, if it was left to wander from a random initial state (or initial states if a Local Beam Search is used [RN03]).

A difference between the proposed method and those proposed by [SN04] and [HHAK06] becomes immediately apparent: the use of the LS allows for a wider variety of constraints and cost function components to be handled, as well as allowing for a more relaxed MIP formulation, making it easier to approach large problems. On the other hand, the constructive phase itself is built over more generic criteria with regards
to the abstractions used. In the proposed method, duties are grouped into clusters according to optimization stage dependent criteria, influenced by the problem constraints and cost function, instead of merely relying on whatever abstractions are known in the specific reality.

4.3 Constructive phase

As described above (see Section 4.1), a major factor in the existence of such a large number of possible states is the amount of combinations between planning elements, namely duties. That said, as can be easily noted, two night duties with similar start/end times will have a very similar influence on the type of roster being created. Following this idea, it is possible to apply a sort of state space relaxation on the problem, where each solution specifies that, on day $i$, some duty $d \in D_c$ is planned, with $D_c$ being the set of all duties that belong to a given duty cluster $c$. This proves to be a powerful tool in controlling the growth of the problem, as long as there is some care in choosing the cluster creation criteria. However, it is still not enough, for a problem of this nature. Solving the complete problem at once would require using very relaxed rules for the cluster creation. This would lead to sub-optimal solutions. Intuitively, even though, on average, duties from two clusters $D_1$ and $D_2$ cannot be planned in consecutive days, some combination within those clusters may be acceptable and, possibly, be in the optimal solution. The more relaxed the rules, the greater the chance of this happening.

The proposed solution to sidestep this obstacle is to solve the problem in successive steps, each corresponding to a sub-problem or abstraction over the main problem. During the resolution, the ratio between the constraints and the allowed duty combinations is carefully managed, as shown in Figure 3.

**Fig. 3.** Constructive process diagram.
At first, when no decisions have yet been taken (minimum constraints), the problem is solved using relaxed clustering rules. From the obtained solution, the day off pattern assignment is taken and given as an additional constraint to the next stage. With a fixed day off pattern, it is now possible to use more refined clustering criteria to obtain the overall cluster assignment. In the final step of the constructive phase, individual duties can easily be assigned to the rosters, subject to the specified clusters and day off pattern.

In order to obtain good results, the clustering rules used in each step are conscious of the step’s goals. For instance, it is important that the clusters considered during the day off assignment take into account criteria that directly influence the planning of days off. In this case, relatively refined criteria are used with regards to critical moments influencing the day off protection times. On the other hand, aspects like the duty duration, may be somewhat neglected. Section 4.5 introduces additional details concerning the clustering rules used throughout the planning process.

Note that, following the generic method definition, the proposed sub-problems may change for different realities. For instance, the days off assignment could be partial (introducing some additional flexibility during the cluster assignment) and other sub-problems could be considered. This obviously depends on the combinatorial nature of the problem, type of constraints and cost function.

4.4 Iterative improvement

Once the initial solution is obtained, the iterative improvement based on a LS is applied, hopefully from a close vicinity of the global optimum. At this point, the search can consider the full complexity of the problem, looking at individual duties and the full set of constraints. In fact, it is not even necessary for the constructive stage to take into account (or, at least, do so with full detail) all the rules or cost function components of the problem. It suffices to consider those with a foreseeable global impact. The iterative improvement stage may then serve as a safety net to handle additional constraints.

Because the state space is, still, very large, care is taken to ensure the convergence of the method in a reasonable amount of time. First, the improvement goals are grouped into categories to which a priority is assigned. The algorithm iterates over the ordered categories and, in each iteration, attempts to improve the situation of the goals in the current category. Only actions that improve on the current situation without increasing the amount of violations of higher priority categories are accepted. It follows that the relative importance of the goals is expressed by the priority of the category they are placed in. The pseudo-code of this solutions is presented in 4, for a state $E$ and a set of goals $G$. The term conflict is used to refer to deviations from a desired goal.

In the implemented solution, an action represents a swap between planning elements. The process of finding the best valid action consists in isolating a problem (goal deviation) and simulating the execution of promising swaps. The best acceptable swap found is applied. This implementation corresponds to a Hill-Climbing Search and, although able to converge quickly towards a maximum, it is relies heavily on a quality initial solution being supplied to be effective.
IMPROVESOLUTION\((E, G)\)

1. for each \(O\) in \(G\)
2.     do
3.         improved ← yes
4.     while improved
5.         do
6.             improved ← no
7.             \(C\) ← COLLECTCONFLICTS\((E, O)\)
8.             for each conflict in \(C\)
9.                 action ← FINDBESTVALIDACTION\((conflict, E, G)\)
10.                if action
11.                       then
12.                           APPLYACTION\((action, E)\)
13.                       improved ← yes
14. return \(E\)

Fig. 4. Iterative improvement pseudo-code.

4.5 MIP Model

This section describes how the Constructive phase is modelled as a Mixed-Integer Program (MIP) to solve an Assignment Problem (AP) with additional constraints. The following parameters are considered:

- \(c \in C\), a cluster of planning elements;
- \(r \in R\), a problem roster;
- \(n \in N_r\), a day in the roster \(r\);
- \(f \in F\), a planning frequency (e.g., Monday, Tuesday);
- \(L_{rf}, r \in R, f \in F\), the set with all days of the roster \(r\) at frequency \(f\);
- \(t_{cf}, c \in C, f \in F\), the number of elements belonging to cluster \(c\) at frequency \(f\);
- \(k_{cr}^{1}, c \in C, r \in R\), the cost of assigning elements belonging to cluster \(c\) to the roster \(r\);
- \(v_{ij}, i, j \in C\), the binary parameter that takes the value 0 if elements of cluster \(i\) and elements of cluster \(j\) cannot be planned in consecutive days;
- \(m_{nc}, r \in R, n \in N_r, c \in C\), the binary parameter that takes the value 1 if the position \(n\) of roster \(r\) must contain an element belonging to cluster \(c\);
- \(g \in G\), a goal to consider in the solution cost;

The decision variable reflects the underlying AP-based formulation:

- \(a_{nc}, r \in R, n \in N_r, c \in C\), the binary variable that specifies that the position \(n\) of roster \(r\) contains an element belonging to the cluster \(c\).

Following the principle of focusing on the main issues, in this model, the cost function components are discretized. Intuitively, this represents the fact that, globally, the
optimization of a few minutes is of little relevance. This is left to the iterative improvement phase. The goals are represented by the following parameter and additional set of variables:

- $k_g^2, g \in G$, the parameter that specifies the cost of violating the goal $g$;
- $w_{grn}, g \in W, r \in R, n \in N_r$, the binary variable that specifies if the goal $g$ is violated at position $n$ of the roster $r$.

The cost function is then described as:

$$\min \sum_{r,n} \left( \sum_c k_c \alpha_{rnc} + \sum_g k_g^2 w_{grn} \right), r \in R, n \in N_r, c \in C, g \in G$$  \hspace{1cm} (1)

It has two major components, one related to the assignment of duties to a certain roster and one based on the custom goals.

The following restrictions specify the fundamental assignment problem constraints:

$$\sum_c \alpha_{rnc} = 1, r \in R, n \in N_r, c \in C$$  \hspace{1cm} (2)

$$\sum_r \sum_n \alpha_{rnc} \leq t_{cf}, r \in R, f \in F, n \in L_{cf}, c \in C$$  \hspace{1cm} (3)

$$\alpha_{rnc} \geq m_{rnc}, r \in R, n \in N_r, c \in C$$  \hspace{1cm} (4)

$$\alpha_{rmi} + \alpha_{r(rem(n+1);0)} \leq v_{ij} + 1, r \in R, n \in N_r, i, j \in C$$  \hspace{1cm} (5)

Constraint 2 ensures that each day has at most one assigned cluster. Constraint 3 specifies the covering upper-bound for the problem clusters. Pre-defined assignments are validated by Constraint 4 (this is used to enforce the days off once they are determined), while legal consecutive assignments are validated by Constraint 5.

### 4.6 Feasibility validation

There are two types of days off: Double (two consecutive days) and Short (and individual day). Double days off may only occur intersecting weekends and a double day off must occur every two weeks. For simplicity, a convention is made regarding the weeks (odd or even) in which a double day off may occur.

The following parameters aid in the formulation of the feasibility constraints:

- $o \in O$, with $O \subset C$, a cluster of short days off;
- $oo \in OO$, with $OO \subset C$, a cluster of double days off;
- $oo_1 \in OO_1$, with $OO_1 \subset OO$, a cluster of double days off starting on a Friday;
- $oo_2 \in OO_2$, with $OO_2 \subset OO$, a cluster of double days off starting on a Saturday;
- $oo_3 \in OO_3$, with $OO_3 \subset OO$, a cluster of double days off starting on a Sunday;
- $n_{roster\_lines_r}$, the number of roster lines in roster $r \in R$;
- $ND_r$, the first indexes of lines that should contain double days off in roster $r \in R$;
- $duration_c, c \in C$, the average duration of the elements belonging to the cluster $c$. 
The following constraints ensure the feasibility of the produced day off pattern (note the use of \( \gamma \) to account for the circularity of the rosters and that removing it would suffice to make the formulations applicable to the planning of individual rosters):

\[
\sum_{n} a_{rno} = n_{roster\ lines}, r \in R, n \in N_r, o \in O
\]

\[
\sum_{n} a_{rno} = n_{roster\ lines}, r \in R, n \in N_r, oo \in OO
\]

\[
\sum_{n' = n+4}^{n+7} a_{ryo} = 2, \gamma = \text{rem}(n', |N_r|), r \in R, n \in ND_r, o \in OO
\]

\[
a_{r(l+5)o} = a_{r(l+6)o}, r \in R, n \in ND_r, o \in OO_1
\]

Constraints 6 and 7 validate the number of single and double days off in the solution (respectively). Constraint 8 validates the number of allowed double days off in every two weeks while Constraint 9 checks for the validity of double days off on Friday (analogous validations for Saturday and Sunday double days off are omitted). Constraint 10 ensures that the maximum number of days without days off is not violated while Constraint 11 tests the maximum number of days off for every 3 days of the roster. These constraints are only present in the first problem, which leads to the day off pattern assignment (they could be in the succeeding ones, but since the days off assignment is complete they would be trivially met). The following are applicable to all sub-problems solved in the constructive phase.

\[
\sum_{n' = n}^{n+13} \sum_{c} a_{rcd\ duration_c} \leq 75:00, \gamma = \text{rem}(n', |N_r|), r \in R, n \in N_1r, c \in C
\]

\[
\sum_{n, c \in N_r \times C} a_{rcd\ duration_c} \leq 7:24 \times n_{roster\ lines}, r \in R
\]

\[
\sum_{n' = n}^{n+2} \sum_{c} x_{rci\ isNight_c} \leq 2, \gamma = \text{rem}(n', |N_r|), r \in R, n \in N_r, c \in C
\]

\[
\sum_{n' = n}^{n+30} \sum_{c} x_{rci\ isNight_c} \leq 9, \gamma = \text{rem}(n', |N_r|), r \in R, n \in N_r, c \in C
\]
\[
\sum_{(n,c) \in N_r \times C} a_{rnc} \times isNight_c \leq 1.35 \times n_{\text{roster lines}}, r \in R
\] (16)

Constraints 12 and 13 validate the work capacity of every two weeks and of each roster, respectively. The remaining constraints all deal with night duties. Night duties are considered undesirable work, thus it is desirable for them to be relatively spaced out. Constraint 14 ensures that no more than 2 night duties are planned consecutively. Constraint 15 establishes the maximum of 9 per 31 days and, finally, Constraint 16 specifies that each roster can have at most 1.35 night duties per roster line.

In the last sub-problem, when assigning individual duties to obtain the solution of the constructive phase, it is necessary to constrain the acceptable duties in each day to the clusters that were determined in the preceding step. First, the specification of which elements can go where:

- \( E \), the set of planning elements in the problem;
- \( U_m \subset E \), the set of planning elements that can be planned in the position \( n \in N_r \) of roster \( r \in R \).

Given the additional parameters above, Constraint 17 below ensures that the assignments obey the specified clusters:

\[
\sum_c a_{rnc} = 1, r \in R, n \in N_r, c \in U_m
\] (17)

Note that the final stage has a very easy time meeting the feasibility constraints, as they will already have been validated in the previous stages leading to the cluster assignment.

### 4.7 Cost function

The cost function has two major components: one related to the day types assigned to each roster (e.g., whether or not the assignment violates the roster’s day type preference) and one related to the day off protection time. The distance between duties is modelled as a hard constraint in that duties that violate it cannot be planned consecutively.

The cost of violating a day off protection time can be seen as growing with the amount of violation. However, as stated above, during the constructive phase the costs must be discretized. In this case, two parcels are considered, one corresponding to a soft and another to a hard violation.

In order to better guide the days off assignment phase, the following criteria are used to specify the clusters (note how they are directed by the soft and hard violation limits):

- \( \text{day type} \), if it is an early, day, late or night duty (every duty falls into one of these categories).
- \( \text{duty start} \in [0:00; 5:00[ \), separates duties that produce a hard violation on the day off protection time if the previous day is a day off;
duty start $\in [5:00; 6:00]$, separates duties that produce a soft violation on the day off protection time if the previous day is a day off;

- duty end $\in [20:00; 23:00]$, separates duties that produce a soft violation on the day off protection time if the next day is a day off;

- duty end $\in [23:00; 02:00]$, separates duties that produce a hard violation on the day off protection time if the next day is a day off.

These goals are incorporated in the cost function through the appropriate configuration of the variable $w$ and corresponding costs $k_1^2$ and $k_2^2$, given the following parameters:

- $\text{isLateDutySoft}_c, c \in C$, the binary parameter that specifies that duties in cluster $c$ violate the soft limit of protection time before a day off;

- $\text{isLateDutyHard}_c, c \in C$, the binary parameter analogous to the previous but for a hard violation.

The following constraints ensure that the variable $w$ is adequately set. Assuming that $n \in N$, $r \in R$, $o \in O \cup OO, c \in C, \gamma = \text{rem}(n + 1, |N_r|)$:

$$w_{1rn} \geq \sum_c a_{rne} \text{isLateDutyHard}_c + \sum_o a_{rmo} - 1 \quad (18)$$

$$w_{2rn} \geq \sum_c a_{rne} \text{isLateDutySoft}_c + \sum_o a_{rmo} - 1 \quad (19)$$

In the second phase, it becomes possible to consider more refined clusters. Good results were obtained with the following rules, which are basically an extension of the previous ones:

- day type, if it is an early, late, night or normal duty (not that every duty falls into one of these categories).

- duty start $\in [0:00; 5:00]$, separates duties that produce a hard violation on the day off protection time if the previous day is a day off;

- duty start $\in [5:00; 6:00]$, separates duties that produce a soft violation on the day off protection time if the previous day is a day off;

- duty start in intervals of 2:00 between 6:00 and 2:00 of the next day, provides further detail on the duties being planned;

- duty end $\in [20:00; 23:00]$, separates duties that produce a soft violation on the day off protection time if the next day is a day off;

- duty end $\in [23:00; 02:00]$, separates duties that produce a hard violation on the day off protection time if the next day is a day off.

- duty end in intervals of 2:00 between 6:00 and 20:00, provides further detail on the duties being planned;

- duty duration in intervals of 1:00 between 6:00 and 9:00, provides further detail on the duties being planned;

---

2 Rules thus specified actually correspond to several rules of the type duty start $\in [6:00; 8:00]$, duty start $\in [8:00; 10:00]$, etc.
5 Results

The proposed method was tested against a solution produced by specialized human planners. The main aspects of interest are (by order of importance):

1. Maximizing the number of planned duties;
2. Minimizing the number of rests between duties with less than 13h (11h is the minimum for a feasible solution);
3. Minimizing the number of complementary rest days;
4. Minimizing the violation of day off protection times;
5. Maximizing the number of duties in rosters with preference over the duties.

Minimizing rest days is of particular interest. A complementary rest day is a day where the resource stays at home with full pay. These days are used to lower the work load of overloaded periods. Using these days is often a tempting solution and sometimes necessary. However, besides meaning that a resource receives pay without working, other resources that are assigned the same overloaded work, or even a part of it, in the short-term (e.g., because the original resource is absent on vacations) will likely also have to be given a rest day, thus propagating the inefficiency.

Table 1 presents a comparison between the results obtained by the proposed method and a method which employs a heuristic search strategy during the constructive phase, on a subset of the original problem, where 394 duties are planned in the 82 lines of 7 rosters. As can be easily seen, the difference between the heuristic constructive phase and the proposed method shows the importance of running the LS from a vicinity of the global maximum, even for a relatively small problem.

<table>
<thead>
<tr>
<th></th>
<th>Heuristic Initial State</th>
<th>Proposed method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run time</td>
<td>0:00:19</td>
<td>3:35:16</td>
</tr>
<tr>
<td>Planned duties</td>
<td>394</td>
<td>394</td>
</tr>
<tr>
<td>Rests under 13h</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Rest days</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Roster average</td>
<td>6:55:30</td>
<td>6:59:40</td>
</tr>
<tr>
<td>Day off protection violations</td>
<td>105:40</td>
<td>52:35</td>
</tr>
</tbody>
</table>

Table 1. Comparison with solutions obtained with an heuristic constructive phase.

A comparison between the proposed method and the solution obtained by human planners on the complete problem (1459 duties on 23 rosters in a total of 304 roster lines) is presented in Table 2. Note how the proposed method was able to obtain solutions without rests under 13h and still improve on every other quality criteria in relation to the human solution. The run time for a problem of this size and impact suggests that finer clustering criteria could have been used, making it possible to further improve on the obtained solution. Other runs were made over partial problems (subset of duties and rosters) and the results were consistent with those presented.
The results were obtained running the prototype on an Intel Core 2 Duo 1.8 GHz with 2GB/Go DDR2 SDRAM. The results for the MIP were obtained with version 11 of ILOG’s CPLEX, without the parallel processing functionality active.

### 6 Summary and Future Work

In this work, a method for solving large-scale rostering problems is proposed that is both efficient and generic, in the sense it could be adapted to different realities while keeping a good overall quality. It can be seen as extending and complementing techniques described in the literature.

The method relies on the creation of a series of abstractions over the real problem, solved using a Integer-Linear Programming and complemented by the execution of a Local Search. In the first (constructive) phase, the focus is on the decisions with greatest impact on the quality and feasibility of the solution. In this case the detail of certain constraints or components of the cost function may be ignored or relaxed. The Local Search is then executed considering the full complexity of the problem.

The proposed method was tested on a very large scale instance of the Cyclic Crew Rostering Problem, taken from the real planning scenario of a European railway company. The method was able to obtain far better solutions than specialized human planners. The possibility of using the Local Search to refine the initial solution and even combining the initial abstractions in different ways introduces the potential for modelling difficult non-linear constraints and cost function components, as long as some appropriate approximation is found in the constructive phase.

One of the next research steps is to apply this method to the planning of individual rosters. Additional improvements may be obtained by exploring advanced clustering algorithms and the parallel processing facility of CPLEX. Better results may also be obtained by introducing a stochastic component to the Local Search algorithm.

### Acknowledgments

This work contains the most significant conclusions of my master thesis, developed at SISCOG and Instituto Superior Técnico. I am grateful to both institutions for the opportunity and the creativity inspiring working conditions.
References


