TOP (2018) 26:383–402
https://doi.org/10.1007/s11750-018-0479-2

ORIGINAL PAPER

A model and computational tool for crew scheduling in train transportation of mine materials by using a local search strategy

Jorge Amaya[1] · **Paula Uribe**[2]

Received: 14 September 2017 / Accepted: 20 April 2018 / Published online: 2 May 2018

Abstract This work introduces a model of the crew scheduling problem for the operation of trains in the mining industry in the North of Chile. The model possesses particular features due to specific regulations with which train operators in mine material transportation are required to comply: every week, a fixed set of trips must be made according to current demand for the transportation of mine products and supplies. In order to balance the loads of the crews in the long term, the proposed model generates an infinite horizon schedule by means of a rotative scheme in which each crew takes the place of the previous one at the beginning of the next week. This gives rise to a medium/large size 0–1 linear optimization problem, whose solution represents the optimal assignment of drivers to trips with the number of working hours per week distributed equally among crews. The model and algorithm have been implemented with a user interface suitable for the remote execution of real instances on a High Performance Computing platform. The transportation company regularly uses this computerized tool for planning crew schedules and generating efficient assignments for emerging and changing operational conditions.

Keywords Crew scheduling · Integer programming · Local search heuristics

This research has been partially supported by Basal project CMM-University of Chile, and by Fondecyt Grant 1130816.

✉ Jorge Amaya
jamaya@dim.uchile.cl

Paula Uribe
pauribe@dim.uchile.cl

[1] Center for Mathematical Modeling and Department of Mathematical Engineering, University of Chile, Av. Beauchef 851, Santiago, Chile

[2] Center for Mathematical Modeling, University of Chile, Av. Beauchef 851, Santiago, Chile

Mathematics Subject Classification 90C09 · 90C10 · 90C27

1 Introduction

Crew scheduling is one of the major phases in crew management in large transportation networks such as railway, bus and airline systems, where technical, legal and time constraints must be taken into account when scheduling drivers and crews. In our particular application, a crew typically consists of two drivers to whom a set of tasks (trips) are assigned from week to week.

The work presented here is concerned with a specific application of a railway company dedicated to the transportation of mining supplies and products. In this case, the greatest interest is in distributing the load in the most balanced way possible between crews so as to guarantee minimal differences between the drivers' salaries while maintaining the weekly load within legal limits stipulated for the industry. The two main contributions of our work are a rotative scheme that produces balanced loads and a special local search strategy used to obtain the solution. Moreover, the model specification is quite general in the sense that the binary variables can be used in a similar way to represent other rotative crew scheduling problems, for example, urban/interurban buses. Another relevant contribution is the development of a software tool that is currently being accessed remotely by the user.

Crew assignment is a classical optimal decision problem (see for example, Ernst et al. 2001, 2004), but the best-known contributions have been made in the context of the air transportation industry. In general, this assignment problem can have a very large number of decision variables which means that solving it entails a high degree of complexity. Frequently, standard branch and bound strategies are not able to solve large instances and many variants of well-known algorithms have been applied in an attempt to deal with these computationally hard problems. For an urban bus system, Desrochers and Soumis (1989) proposed a column generation approach to solve the transit crew scheduling problem which is decomposed into two sequential stages: a set covering problem and a shortest path problem with resource constraints. For the aircrew rostering problem, Gamache et al. (1999) used a generalized set partitioning model together with a method using column generation which is adapted to take advantage of the structure of the problem. They claim that this method is capable of solving very large scale problems with thousands of constraints and hundreds of sub-problems. A hybrid column generation approach for the urban transit crew problem was studied by Yunes et al. (2005). There, the problem was divided into two stages: crew scheduling and crew rostering, with each being solved separately as well as combining mathematical programming and constraint logic programming with column generation. A similar approach was used by Nishi et al. (2014), where a two-level decomposition for solving a railway crew rostering problem is presented. This decomposition combined an efficient branch and bound algorithm to solve the master problem (assignment of duties to rosters) and a general purpose solver for the lower-level problem (generation of a feasible sequence of rosters). The development and implementation of an integer optimization model to solve disruptions to an operating schedule in the rail industry was described by Walker et al. (2005). The article

presents favorable results for both the combined train/driver scheduling model and the real-time disruption recovery model. In contrast, Abbink et al. (2007) used iterative partitioning to handle large-scale crew-scheduling instances. Lagrangean relaxation combined with subgradient optimization was applied by Beasley and Cao (1996). Decomposition and relaxation strategies were used by Vaidyanathan et al. (2007) for the solution of a multicommodity network flow problem representing the railroad crew assignment. The multicommodity flow was again addressed by Mesquita et al. (2015) for a bus driver rostering problem incorporating a day-off pattern, where the authors propose 3 different MIP models and developed a heuristic consisting of sequentially solving sub-problems defined using hierarchical ordering of the decisions. Heuristic approaches, such as simulated annealing and genetic algorithms, were proposed by Emden-Weinert and Proksch (1999), Jian and Chou (2010) and Levine (1996), for both airline and train crews. Sanders et al. (1999) apply high performance Integer Optimization to the practical solution of the crew scheduling problem. They use heuristics based on Lagrangean relaxation and a sequential active-set strategy. The problem of finding the minimum number of crews required to carry out a given set of duties is presented by Şahin and Yüceoğlu (2011), where a sequential approach is developed in which the minimum capacity is first calculated without a day-off requirement and then additional crew capacity is added to handle the day-off requirement. The authors also developed an integrated approach consisting of a flow-problem solution with a day-off requirement that turned out to be superior in almost every case. Finally, a very practical contribution is due to Jütte et al. (2011), where they describe a useful software developed for a freight carrier company, based on a set-covering-type formulation and a column-generation solution technique.

In general terms, our crew scheduling problem consists of a given number of weekly trips that have to be assigned to a given set of crews. Any method proposed for solving the problem must first provide a feasible solution composed of a rotative weekly schedule consisting of a sequence of fulfilled trips (that is, trips with a crew assigned), and second, an allocation of crews to weeks. Trips and the number of crews are known a priori. The objective is to obtain a feasible and balanced schedule for crews (in terms of total hours served in a week) that respects legal and operational constraints.

The concept of balanced loads used here is not new and is natural in many cases, but there are few related models in the literature. In Beliën et al. (2013), Beliën et al deal with the problem of constructing the workforce schedules of an aircraft maintenance company. In their model they use a certain notion of rotation as we propose here, but they consider staffing levels and time windows for jobs. In our case, we impose linking conditions to the transition between two successive weeks and the modeling and solving approaches are completely different. Another point of view for this problem can be found in the theory and practice of inequity aversion problems, where equity is the main concern and it can be competitive with other notions of efficiency (see for example, Karsu and Morton 2015), but this framework is rarely used in the context of railway crew scheduling.

The work presented here comes from a specific application of a Chilean railway company. The company distributes supplies from coastal ports to mining concerns in the North of the country and transports mining products for export on its way back to the ports. For this case, primary interest is in distributing the load between m crews

as equally as possible while maintaining the weekly load within certain legal bounds. The problem solution must also provide an answer composed of a rotative weekly schedule, in which after m weeks, every crew will have met the program for each week. Moreover, this strategy delivers an infinite horizon schedule, however, new schedules can be made as often as desired. Due to the rotative scheme, even when the workload of the first week of a given diagram generated by this model deviates a lot from the rest of weekly total hours, we observed in our numerical simulations that the accumulated average after 4 weeks becomes quite stable.

The general solution approach is given in three sequential steps. Firstly, a feasible solution is obtained, which is equivalent to a schedule where every trip is covered, but the working-hours load is not necessarily balanced among weeks. Secondly, a local search heuristic is used to improve the initial feasible solution, by balancing the weekly crew loads. Finally, specific crews are assigned to the scheduled weeks, taking into account the initial conditions of crews, in terms of current location, immediately preceding load and hours-of-rest requirements.

The solution was packaged into Java-based software to allow the user to easily interact with data and simulate different scenarios via remote execution using High Performance Computing (HPC) resources for large-scale instances.

The rest of the paper is organized as follows: The basis for the crew-management planning problem and the integrated optimization model for crew scheduling are set out in Sect. 2. The algorithmic strategy for generating feasible and balanced schedules and the flow model for assigning specific crews to weeks, are presented in Sects. 3 and 4. In Sect. 5 we describe the implementation of the software to enable the user to interact with the computation modules through a user interface. Finally, in Sects. 6 and 7, we present some case studies and the main conclusions, summarizing our numerical experiments on real instances.

2 The mathematical model and the solution strategy

The set of constraints must encapsulate the legal regulations and the specific labor agreements and contracts between the company and the workers' unions.

1. **One trip, one crew**. Each trip must be assigned to one and only one crew, but one specific crew can of course be assigned to several trips during the week.
2. **Overnight legal rest**. For each 7-day window there must be at least 1 legal rest. In our specific case study, a legal rest corresponds to a rest period which is represented by a fictitious trip of 33 h, beginning at 9 p.m. and ending at 6 a.m. 2 days later. The starting time and duration of this rest period are parameters of the general model introduced here. See Constraint (6).
3. **Inter-trip rest**. Between a couple of consecutive trips a time window called an *inter-trip* rest must be imposed. The duration of this window is given by the labor-regulation laws, but in this specific case study this parameter is fixed between 8 and 12 h. This is a relatively long inter-trip rest period, but in our application, most trips take from 6 to 12 h to complete. See Constraints (3) and (4).
4. **Sunday rest**. There are rest regimes of 0, 1 or 2 Sundays every 4 weeks. The Sunday rest, starting at 0:00 and ending at 24:00 (but this can be parameterized),

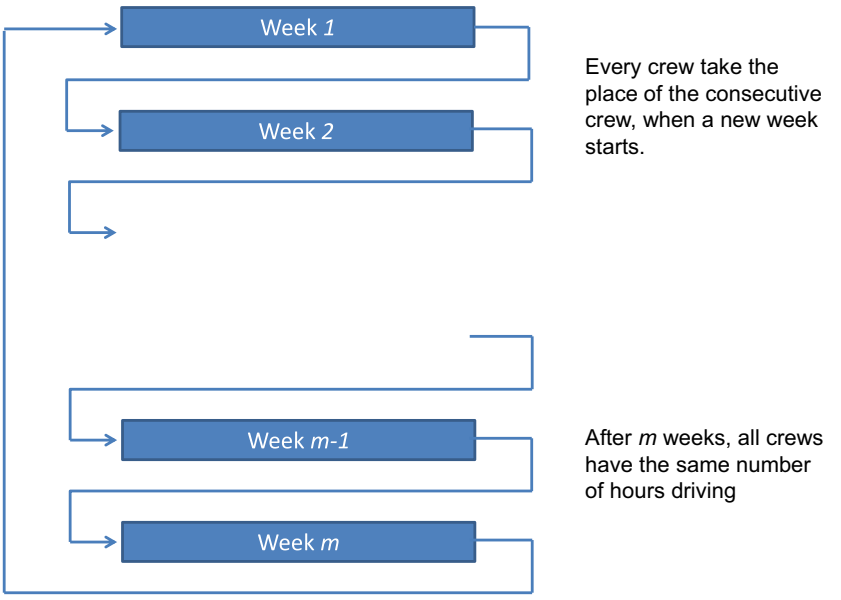

Fig. 1 The rotation scheme

must be assigned according to one specified regime. If a legal rest falls on Sunday, then it can be considered as a Sunday rest. See Constraint (7).

5. **Origin/destination**. For a given assigned crew, the origin of a trip must be the final destination of the preceding trip. This is also included in the exclusion Constraints (3) and (4).
6. **Rotation**. An innovative and simplifying idea proposed for this company consists of imposing a rotation scheme where a crew i follows the schedule of crew $i + 1$ in the next working week. In this manner, after m weeks (m being the number of crews), all crews have followed all the schedules, which implies that the number of hours worked by all the crews is the same in the long run. This is shown in Fig. 1 and represented by Constraint (5).
7. **Consecutive trips**. There are pairs of trips that together form a complete round trip. In these cases, it is required that an outbound trip be followed by the corresponding inbound trip. See Constraint (8).

Concerning the objective function, assuming that all trips can be served by a crew, the most critical issue for this application is to schedule the workload among crews as balanced as possible. This means that the variance of the number of hours per week could be used as the minimizing criterion, but this is a non-linear function. To avoid this difficulty we prefer to use a different metric that can be formulated in a linear form.

Let $\mathcal{V}_0 = \{v_1, \ldots, v_n\}$ denotes the set of train trips in a week. We assume that these trips are regular, in the sense that the same schedule is repeated every week. In our model, we consider the rest days (legal and Sunday) to be special types of virtual trips, as explained later in this section.

Let $\mathcal{V} = \mathcal{V}_0 \cup \{\texttt{virtual trips}\}$ denote the set of all trips. Each trip in $\mathcal{V}$ is characterized by a vector of attributes or parameters which we take as the given input data for the model. These are: starting time (day, hour, minute), travel duration, initial station or origin and final station or destination. So, we assume that the following information is known (for any trip $v \in \mathcal{V}$):

- I_v and F_v denote the initial station and the final destination of v, respectively;
- (h_v, m_v) denotes the starting time (hour, minute) of trip v, where h_v and m_v are integer values satisfying $0 \leq h_v \leq 23$ and $0 \leq m_v \leq 59$;
- $(\Delta h_v, \Delta m_v)$ analogously denotes the duration for trip v in hours and minutes; and
- $(\bar{h}_v, \bar{m}_v)$ similarly denotes the arrival time for trip v.

It can be deduced that

$$\begin{aligned} \bar{h}_v &= h_v + \Delta h_v + \lfloor (m_v + \Delta m_v)/60 \rfloor \mod 24 \\ \bar{m}_v &= m_v + \Delta m_v \mod 60 \end{aligned}$$

There is also a general attribute: the rest regime, $R = 0,\ 1,\ 2,\ 3$, indicating the number of free Sundays in each 4-week interval for every crew. Note that $R = 4$ is in general forbidden, because in that case the problem would become infeasible if at least one trip must run on Sunday.

We now come back to the two types of virtual trips associated with crew rests. Legal (overnight) rests are interpreted as (possibly, but not mandatory) trips running every day $k \in \mathcal{D} = \{1, \ldots, 7\}$ and Sunday rest only runs for $k = 7$. An overnight legal rest for crew i is then a virtual trip $v = v_{LR}$ whose attributes are

$$\begin{aligned} (h_{LR}, m_{LR}) &= (21, 0) \\ (\Delta h_{LR}, \Delta m_{LR}) &= (33, 0). \end{aligned}$$

That is, the starting time is 21:00 on day k, ending at 06:00 on day $k + 2 \pmod 7$. Also, for this particular kind of fictitious or virtual trip, the origin and final destination are the same.

The Sunday rests $v = v_{\mathrm{SR}}$ are similarly defined, but only (possibly) running on Sunday:

$$\begin{aligned} (h_{\mathrm{SR}}, m_{\mathrm{SR}}) &= (0, 0) \\ (\Delta h_{\mathrm{SR}}, \Delta m_{\mathrm{SR}}) &= (24, 0). \end{aligned}$$

All these values, starting time and duration, can be parametrized.

In the rest of this section, we specify the variables and constraints associated with our model. We use $\mathcal{D}_v \subseteq \mathcal{D}$ to denote the set of days on which trip v takes place.

Let x_{ivk} be an integer 0–1 variable indicating if crew $i \in \mathcal{T} = \{1, \ldots, m\}$ is allocated to trip $v \in \mathcal{V}$, starting on day $k \in \mathcal{D}_v$, that is,

$$x_{ivk} = \begin{cases} 1 \text{ if crew } i \text{ is allocated to trip } v \text{ on day } k \\ 0 \text{ otherwise} \end{cases}$$

With these variables, we can now specify the constraints of the optimization problem to be solved.

2.1 The optimization model

Since the idea is to achieve a balanced number of working hours per week for the crews, we use the objective (balanced crew scheduling):

$$\text{(BCS)} \quad \min \quad (z^+ - z^-),$$

where z^+ and z^- are (positive) integer variables representing the minimum and maximum weekly workloads of the crews, respectively, as explained below.

The requirements can be expressed through a set of (in)equalities representing the constraints of the mathematical model.

- **Balanced weekly loads**. From the definition of z^+ and z^- we must impose the constraint:
$$z^- \leq \sum_{v \in \mathcal{V}_0} \sum_{k \in \mathcal{D}_v} \Delta_v x_{ivk} \leq z^+ \quad \forall i \in \mathcal{T}, \tag{1}$$
where Δ_v is the duration of trip v.
- **One trip, one crew**. Each real (not virtual) trip must have one and only one crew, so we impose the following constraint:
$$\sum_{i \in \mathcal{T}} x_{ivk} = 1 \quad \forall v \in \mathcal{V}_0,\ k \in \mathcal{D}_v \tag{2}$$
This constraint does not apply to virtual trips, because either no crew or several crews may be resting at the same time.
- **Incompatibility between two trips.** Let us define the compatibility index for a pair of trips. If v, v' are two trips in days k and k', respectively, then we define a parameter $\eta_{vkv'k'}$ by
$$\eta_{vkv'k'} = \begin{cases} 1 \text{ if } (v,k) \text{ is compatible with } (v',k') \\ 0 \text{ otherwise} \end{cases}$$
This compatibility index is calculated considering the time of arrival/departure and the origin/destination of the two trips. For a precise definition of $\eta_{vkv'k'}$, we define the parameter:
$$H(v,k) = 24(k-1) + h_v + \frac{m_v}{60}$$
which is the starting time of trip (v,k), represented by a value in the real interval $[0,\ 168]$. Similarly, we define
$$\bar{H}(v,k) = H(v,k) + \Delta h_v + \frac{\Delta m_v}{60},$$

the finishing time of trip (v, k). This value can be greater than 168.
Then, the array η is calculated as follows. For real trips (v, k) and (v', k') satisfying $k' \geq k$ and $\bar{H}(v, k) \leq H(v', k')$, and the parameter δ denoting the minimum rest time between two successive trips (which is typically between 8 and 12 hours for our application), we define the parameter:

$$\eta_{vkv'k'} = \begin{cases} 1 \text{ if } \quad \bar{H}(v, k) + \delta \leq H(v', k') \quad \wedge \quad \{I_{v'} = F_v \vee k' > k + 1\} \\ 0 \text{ otherwise} \end{cases}$$

The incompatibility constraint is then expressed as follows:

– For different days $k < k'$:

$$x_{ivk} + x_{iv'k'} \leq 1 \quad i \in \mathcal{T}, \quad v, v' \in \mathcal{V}_0, \quad \eta_{vkv'k'} = 0 \tag{3}$$

– For the same day $k = k'$:

$$x_{ivk} + x_{iv'k'} \leq 1 \quad i \in \mathcal{T}, \quad v, v' \in \mathcal{V}_0, \quad v \neq v', \quad \eta_{vkv'k'} = 0 \tag{4}$$

The only exception to this time incompatibility is permitted on the virtual trips representing crew rest days. This means that a *rest trip* is compatible with all (real or virtual) trips.

– **Crew rotation.** In order to impose the trip incompatibility constraint taking into account that crew i takes over the schedule of crew $i + 1$ the next week, we write

$$x_{iv7} + x_{i'v'1} \leq 1 \quad \forall i \in \mathcal{T}, \quad v, v' \in \mathcal{V}, \quad \eta_{v7v'1} = 0 \tag{5}$$

with $i' = i + 1 \mod m$.

– **Overnight legal rest.** The legal rest must be assigned at least once in each 7-day window, so we impose the constraint

$$1 \leq \sum_{j=k}^{k_m} x_{ivj} + \sum_{j=1}^{k-1} x_{i'vj} \quad v = v_{LR}, \quad i \in \mathcal{T}, \quad i' = i + 1 \mod m, \quad k \in \mathcal{D}, \tag{6}$$

where $k_m = \min(7,\ k+6)$. Evidently, for $k = 1$, the second term on the right-hand side does not apply.

– **Sunday rest.** The Sunday rest regime indicated by the R attribute determines the number of free Sundays in a group of 4 consecutive weeks. The corresponding constraint is written as follows:

$$\sum_{j=i}^{\min(i+3,m)} x_{jv7} + \sum_{j=1}^{i+3-m} x_{jv7} \geq \mathcal{R} \quad v = v_{\mathrm{SR}}, \quad \forall i \in \mathcal{T}. \tag{7}$$

– **Consecutive trips.** In some cases, the decision-maker imposes that a given pair of trips (v, k) and (v', k') must be served by the same crew (for example, round

trips). To impose these conditions we define the parameter

$$\gamma_{vkv'k'} = \begin{cases} 1 \text{ if } (v,k) \text{ and } (v',k') \text{ must be served by the same crew} \\ 0 \text{ if } (v,k) \text{ and } (v',k') \text{ can be served by different crews} \end{cases}$$

and the constraint is then written as

$$x_{ivk} = x_{iv'k'} \quad \forall i \in \mathcal{T}, \quad \gamma_{vkv'k'} = 1 \tag{8}$$

To avoid infeasibility, the data must satisfy the condition $\eta_{vkv'k'} = 1$ for all pair of trips (v,k) and (v',k'), such that $\gamma_{vkv'k'} = 1$. In other words, Constraint (8) can only be applied to compatible trips.

3 The adapted local search strategy

In practice, the problem formulated above is hard to solve, specially due to Constraint (2), which forces every trip to be assigned to a crew. The computational effort, in terms of execution time, can be reduced if Constraint (2) is relaxed to

$$\sum_{i \in \mathcal{T}} x_{ivk} \leq 1 \quad \forall v \in \mathcal{V}_0, \quad k \in \mathcal{D}_v, \tag{9}$$

which permits some trips to be left without a crew being assigned to them. Then, the optimization problem is now defined by constraints (3)–(9) and the objective function of this (feasible crew scheduling) problem is as follows:

$$(FCS) \quad \max \sum_{i \in \mathcal{T}} \sum_{v \in \mathcal{V}_0} \sum_{k \in \mathcal{D}_v} x_{ivk}, \tag{10}$$

which corresponds to maximizing the number of fulfilled trips. If (BCS) is feasible, then at the optimal solution of (FCS), Constraint (9) becomes active and the problem (FCS) reaches the maximum value at the total number of trips to be served every week, that is,

$$\sum_{v \in \mathcal{V}_0} |\mathcal{D}_v|$$

In fact, the objective function (10) represents a sort of phase I of Linear Programming, which looks for a feasible solution to initialize the optimizing part of the algorithm.

Given the simplified formulation above, one can solve the problem of finding a balanced trip allocation by combining the mathematical model with a heuristic routine which implements local search. The local search routine consists of 3 stages.

– **Stage 1:** We perform Phase I, whose aim is to find a feasible solution of problem (BCS). For this, we find an optimal solution to the problem (FCS), under Constraints (3)–(9), where we seek to fulfill every trip. The relaxation means that

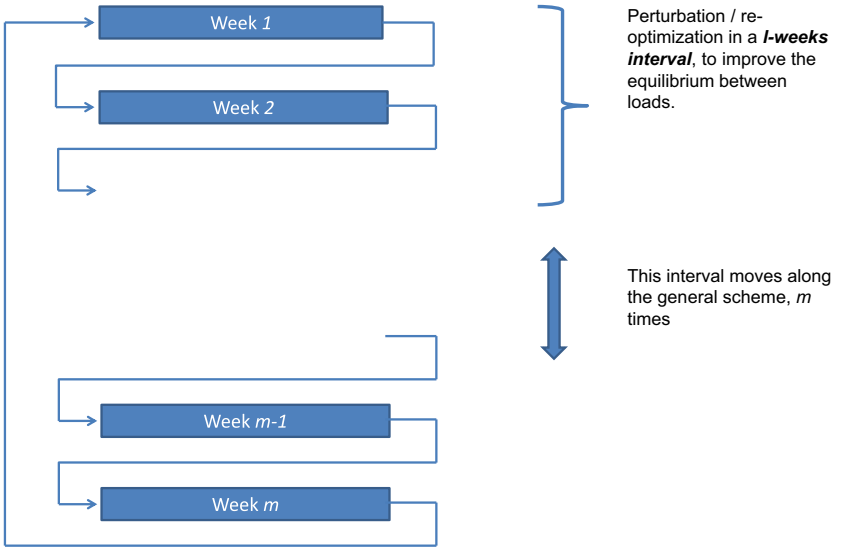

Fig. 2 The local search

Constraint (1) is not included and Constraint (2) is replaced by the relaxed version (9).
If the optimal value (10) reaches the upper bound $\sum_{v \in \mathcal{V}_0} |\mathcal{D}_v|$, the solution is a feasible solution for the main problem (BCS). The case in which (10) doesn't reach this upper bound means that some trips cannot be served (i.e. the problem (BCS) is infeasible) and the number of crews (m) should be increased.

- **Stage 2:** The optimal solution of the previous stage enables a feasible diagram to be generated, that is to say, a weekly schedule for drivers for the first week, as shown in Figs. 1 and 4. This diagram is fragmented into windows of a few weeks as shown in Fig. 2 (in practice, these windows must have a maximum size of around 10 weeks). The local search-based heuristic is an iterative routine that takes a window, fixes the variables outside of it and leaves the variables within free for re-optimization by applying the model for finding a balanced solution, using objective function (BCS). This process is repeated m times, running through all weeks and solving a low-dimensional sub-problem on each iteration, as shown in Fig. 2. Moreover, we include the least and most loaded weeks in each of these sub-problems with the aim of improving the objective function (BCS).
- **Stage 3:** Starting from the last feasible solution (*warm start*), one last re-optimization is performed by releasing all variables and applying the balanced solution model (BCS) to the whole diagram, with a time limit constraint in order to ensure the process will end within a reasonable execution time.

This approach takes advantage of the fact that solving a problem using *warm start* strategies decreases the execution time, since the number of feasible branches is immediately reduced in the Branch and Bound algorithm. This, combined with the strong reduction of complexity when multiple sub-problems are solved instead of a single

big problem, greatly reduces the execution time and provides very balanced solutions, as we can see in the case studies.

The main idea underlying our general approach is to produce balanced schedules in terms of weekly workload. This can be done in two ways. First, by fixing a time horizon and building a plan minimizing the differences of workloads between crews throughout that horizon. Another possibility (our choice) is to produce a general schedule for all the crews for the first week (even an unbalanced solution) and then converge to a balanced scheme by rotating crew assignments. The application of our tool to real cases shows that, in practice, the crews tend to obtain more balanced workloads in a short time during the first 3–4 weeks, even if their workloads are highly variable at the beginning. For this reason, at phase I we focus on obtaining a feasible solution for the first week, regardless of the dispersion of the workload. The second phase seeks to improve the imbalance in this first week, for which we apply the local search strategy described in Sect. 3. Evidently, phase I can also be performed by minimizing the auxiliary objective function in (10) combined with Constraint (9) written in $\geq$ form. In our practical applications, this change does not produce any essential difference with respect to the approach we applied here.

4 The crew assignment problem

The previously described model permits an optimally balanced trip diagram to be found, but it does not include the identification of crews. For the crew assignment, we propose considering the previous model as an input, which provides a solution but without identifying the specific crew to be assigned to each weekly diagram. Then, the problem in this section is to find an optimally balanced diagram.

Let $i \in \mathcal{T}$ be a given crew and $j \in \mathcal{T}$ be a week of the weekly diagram given by the main crew scheduling model. We let w_{ij} be the weight of crew i to be assigned to week j. This term can be proportional to the difference between the number of hours accumulated by the crew i in the previous week and the workload of week j.

We use the variable

$$y_{ij} = \begin{cases} 1 \text{ if crew } i \text{ is assigned to week } j \\ 0 \text{ otherwise} \end{cases} \tag{11}$$

This means that each crew is assigned to one and only one week of the diagram, and each week is assigned to one and only one crew. We also define a bipartite graph whose vertices can be divided into two disjoint sets: the set of crews and the set of weeks. The set $\mathcal{A}$ of oriented arcs connecting crews to weeks is defined as follows:

$$(i, j) \in \mathcal{A} \Longleftrightarrow i \text{ is } \textit{compatible} \text{ with } j$$

Compatibility here means that a given crew, say, i, can be effectively assigned to a given week j. This can be expressed by the following three conditions:

– **Rest hours.** Let (v_i, k_i) be the last trip served by crew i in week $j-1$ and (v_j, k_j) the first trip of the current week j. Then *compatibility* imposes

$$\bar{H}(v_i, k_i) + \delta \leq H(v_j, k_j) + 168 \tag{12}$$

That means that trip (v_j, k_j) must start at least δ hours after the end of trip (v_i, k_i).

– **Feasible location of the crew.** The current location of the crew i must be equal to the starting location (origin) of the first trip in the current week j. That is to say, let L_i be the current location of crew i and I_v be the origin of trip v_j, then we impose

$$L_i = I_v \tag{13}$$

– **Legal rest day.** The last legal rest day taken by crew i must satisfy the legal rest day condition with respect to the current week j (one day off in every 7-day interval). If we let $k_i \in \mathcal{D}$ be crew i's last rest day and $k_j \in \mathcal{D}$ the first rest day scheduled in week j, then

$$k_j \leq k_i \tag{14}$$

The objective of this problem is the following:

$$\max \sum_{(i,j)\in\mathcal{A}} w_{ij} y_{ij} \tag{15}$$

Given that the number of weeks of the diagram and the number of available crews are equal, this problem can be interpreted as finding an optimal one-to-one assignment between crews and weeks. Then the constraints are as follows:

$$\sum_{i\,/\,(i,j)\in\mathcal{A}} y_{ij} = 1 \quad \forall j \in \mathcal{T} \tag{16}$$

and

$$\sum_{j\,/\,(i,j)\in\mathcal{A}} y_{ij} = 1 \quad \forall i \in \mathcal{T}. \tag{17}$$

The model (15)–(17) that deals with the assignment of crews to the scheduled weeks is a simple bipartite graph, where source nodes are represented by crews and destination nodes by the scheduled weeks. A one-to-one assignment is then performed. The feasibility depends on the initial conditions of crews, mainly the current location, the accumulated hours worked and the last legal rest day. The costs w_{ij} of the arcs correspond to the square of the difference between the normalized coefficients of the accumulated crew load and the load for the scheduled week. Thus, the objective function is to maximize the sum of the arcs weighted by their cost, which forces highly loaded crews to be assigned to lightly loaded weeks and vice versa. This is a medium size optimization flow problem whose solution is easy to obtain, in comparison with the computing time for the main scheduling problem.

5 The implementation

Our general strategy consists of three successive stages:
Phase I. Solution of the relaxed problem [in which Constraint (9) is used instead of (2)] to find a feasible solution, using the auxiliary objective function (10).
Phase II. Improvement of the balance between crews by applying the local search strategy, according to the optimization strategy described in Sect. 3.
Phase III. Given the efficient schedule generated in the previous stages, the crew assignment is performed according to the driver assignment flow model presented in Sect. 4.

The optimization models and heuristic routines were written in AMPL, a specialized programming language for linear models that provides enough flexibility for a large range of operations. All the developments for this application were packaged into a Java-based software package with a simple user interface that allows for easy interaction and the creation of several simulation instances with the possibility of running the models using remote HPC resources. The main features of this software are to allow the user to solve different problems for various scenarios, changing parameters such as the number of crews, trip attributes and time limits. It also allows data files to be easily uploaded and output solutions to be downloaded in various formats. The user can choose between local execution using an open-source optimization solver for small problems and a remotely hosted, proprietary optimization solver that employs HPC resources for large-scale instances.

The interface follows a sequence of stages whenever a new instance is executed, as described below:
Read/Transform data to AMPL language. Taking the data uploaded by the user, new data files are generated in AMPL, so they can be read by the optimization solver.
Connect to the remote HPC platform via SSH. If the remote execution option is selected, communication must be established with the remote server that hosts the optimization solver in the first place. In this case we have used SSH (Secure Shell) as the communication protocol.
Send data to High-Performance computer. Once communication has been successfully established, the data in AMPL are sent to the remote computer.
Trigger the execution routine. At this stage, the interface sends the instruction which causes the remote system to start solving the problem. While the program is being executed, the interface waits for a response, polling the remote server at regular intervals for the availability of the solution file. Once the solution is available, the interface retrieves the file, translates it into XML format and displays it on screen to be inspected by the user and/or exported.

Some screen shots of the software can be found in Figs. 3 and 4. Figure 3 shows an example of a set of trips with their corresponding data. Each row corresponds to a trip characterized by trip code, starting time, duration, origin and destination, and days of the week in which the trip must run. This information is provided in an Excel file and can be directly modified on screen. Figure 4 presents the schedule found for the same set of trips. Each row corresponds to a programmed week (without crew identification), starting and final time, total number of hours served in the week, and other operational information. We note that at the end of the schedule, a report of the

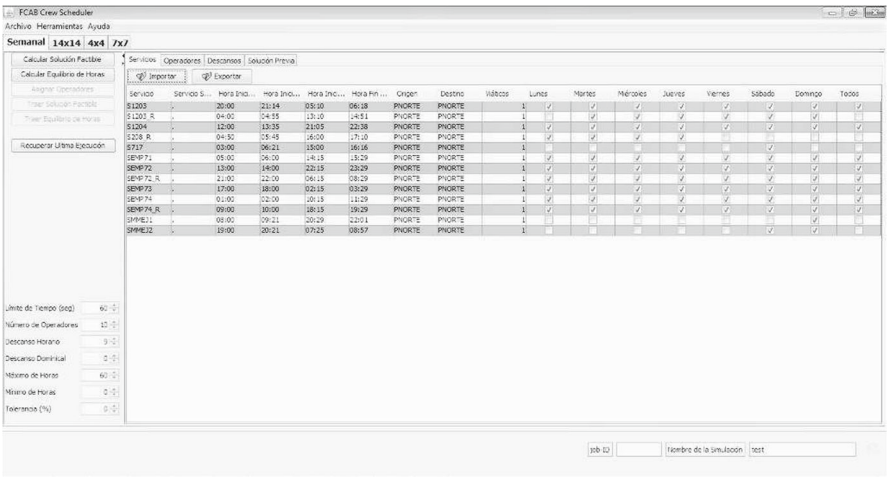

Fig. 3 Example of input trip data for the computer tool

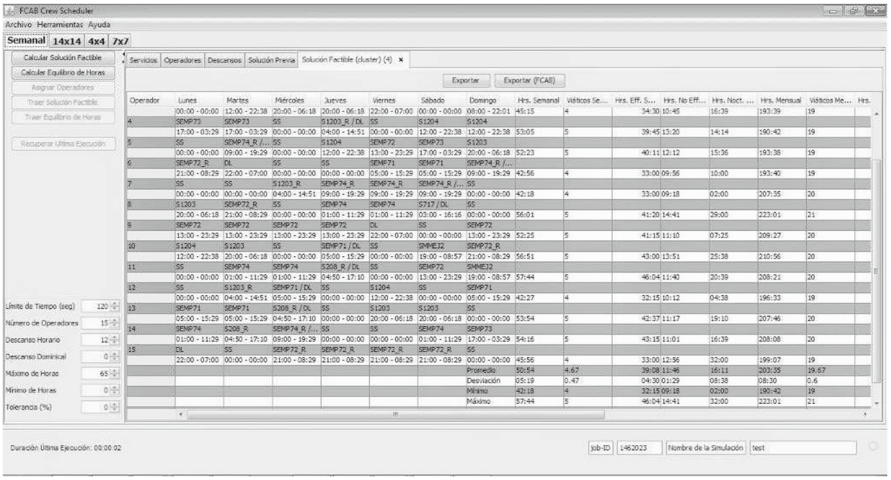

Fig. 4 Example of regular schedule from the computer tool (output diagram)

main statistics of the solution is displayed and the user can export the results to an Excel file. Another possible diagram to visualize the solution is presented in Fig. 5, corresponding to a case with 55 crews. Colors represent the different trips, gray boxes correspond to legal rests and white spaces, to idle times. The total number of working hours in the week is shown at the right-hand side.

The interface also enables the 4×4, 7×7 and 14×14 diagram programming, but the methodology is not described in this article. In an $N \times N$ regime, the crew works with no overnight legal rest for N days, and then rests for the next N days. This is a non-rotative regime; therefore, the optimization problem is less complex to solve compared to that of the regular schedule. The mathematical model for the $N \times N$ case

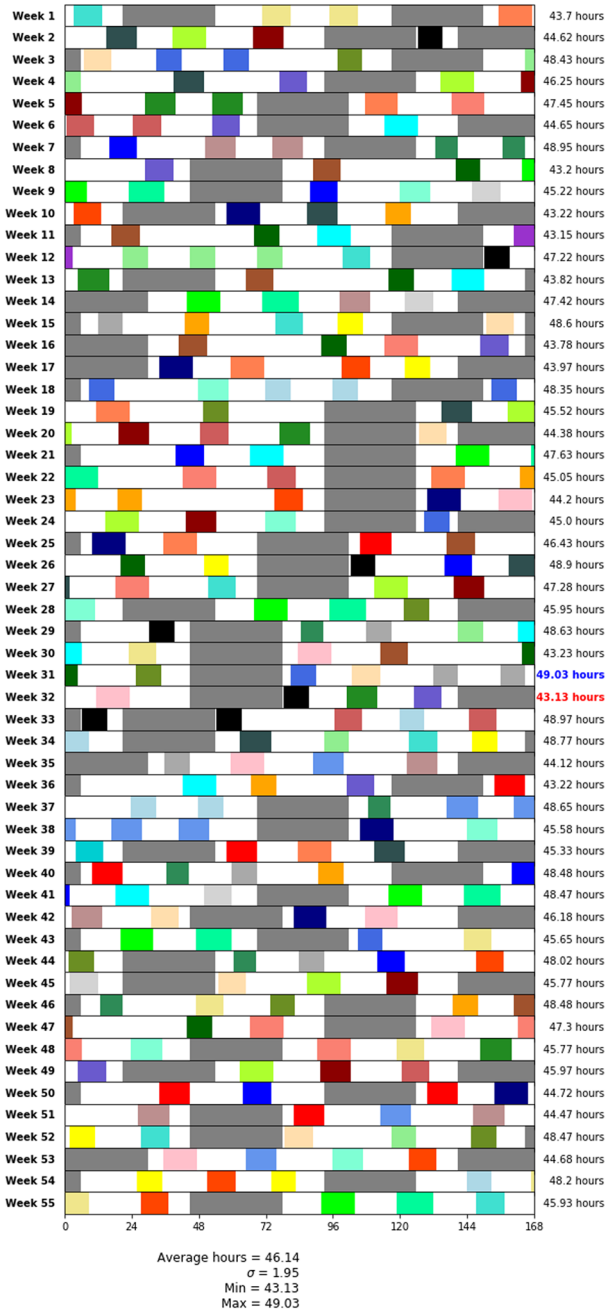

Fig. 5 The schedule diagram

Fig. 6 Main lines of the railway network

is a simplified version of our main model, the essential difference being that rotational, overnight legal rest and Sunday rest constraints are omitted.

6 Case studies

In this section, we present numerical results obtained using the general strategy presented in Sect. 5. All instances were run using a commercial license of solver Gurobi 7.5.1 and AMPL for Gurobi, on an Intel Xeon E5-2660 v2 processor with 10 cores and 48 Gb of RAM. We do not report here results on the assignment problem (Sect. 4) because this is a simple flow problem that can be trivially solved in all our instances. The potential of our approach is essentially in the initial feasible solution and optimizing phases.

In Fig. 6 we show the main lines of the rail network and stations in Antofagasta region (126,000 km^2). It can be seen in the picture that there are two main branches of the network that meet at the coastal city of Antofagasta, where a major port in northern Chile is located. Along the path of the network, there are different courses that cover various geographic zones with variable extension and operative characteristics. From

Table 1 Cases description

ID	Number of binary variables	Average weekly load (h)
1	125	44.3
2	852	57.8
3	1818	43.1
4	1872	49.3
5	13,860	46.1
6	13,986	49.0
7	15,030	44.3

Table 2 Numerical results

ID	Feasibility and Local search					Balanced solution	
	Feasibility		Local search		Total time (s)	σ	Total time (s)
	σ	Time (s)	σ	Time (s)			
1	2.9	0.1	2.0	56.9	56.9	2.0	5.8
2	3.0	9.1	2.2	8.4	17.5	2.2	22.5
3	3.0	19.0	1.4	561.0	580.0	1.4	7199.2
4	2.5	76.5	1.3	660.5	737.0	–	> 7200
5	3.0	108.8	1.4	1368.4	1477.2	2.4	7198.9
6	3.1	779.6	1.9	957.8	1737.4	–	> 7200
7	2.8	1428.7	1.8	1449.3	2878.0	–	> 7200

(–) no solution found in 2 h running

test experiments and practical validation on real cases, we determined that execution time increases with the number of variables and also, this effect is specially critical when the model for finding a balanced solution is applied.

Tests using the heuristic algorithm have shown it is possible to achieve balanced schedules in half the time taken by the balanced solution model and this result can be improved by a factor of 10 when the heuristic algorithm is applied to medium-sized problems. Below in Table 1, we describe some small, medium and large-size cases we consider for testing. All cases were executed on the same computer with the same software settings and legal constraints (values for inter-trip rest and Sunday-rest regimes).

Table 2 shows a comparison of results for the different real-use cases described above. Execution time of the two-phase algorithm and direct resolution are compared for several use cases (see columns in bold). In most cases, the two-phase algorithm turns out to be significantly better in terms of time and balance of hours than the instances in which no heuristic is used.

In the table, Feasibility refers to Phase I, that is to say, the first feasible solution found in which weekly working hours are not necessarily balanced among crews. The actual objective function for this phase corresponds to the maximization of the total number

of trips having an assigned crew but, for display purposes, the standard deviation is indicated in the table for each use case. Local search (Phase II) corresponds to the heuristic stage where the feasible solution found in Phase I is used as a warm start and improved to obtain a more balanced schedule. The third set of results is obtained by executing the Balanced model by direct resolution, without adding any heuristic.

In general, results show that, in terms of the execution time, it is much faster to execute the two-phase algorithm than calculating a balanced schedule directly, except for very small problems, where the sum of the execution times of phases I and II far exceeds the balanced-hours model execution time. This is because for small problems with less than 10 crews (for example, Case 1), the number of variables and constraints can be handled perfectly by any good software; thus, the heuristic option does not seem very useful in this case. At any rate, we are talking about execution times on the order of minutes, which is perfectly tolerable.

In terms of balancing the number of hours worked per week, results show that in all cases, the heuristic routine achieves equal or smaller deviations in half the time compared to the balanced-hours model. Additionally, we can see that for "demanding" cases, (see Cases 4, 6 and 7), the balanced-hours model is not able to find a solution in 2 h. We could say, then, that our heuristic performs better in large-scale problems and represents a substantial improvement when compared to "brute-force" solutions.

7 Conclusions

In this article, we have tackled a real problem on crew-schedule modeling, with the novel feature of including a rotational constraint that delivers balanced workloads among crews while generating a reusable and infinite time horizon schedule. We solved it by using a combination of standard integer programming and a local search strategy, which has shown very good behavior in the instances examined.

The balanced-hours model can be slightly modified in order to generate a balanced schedule in a shorter execution time by relaxing the one-trip one-crew constraint and adding upper and lower bounds to the total weekly hours for all crews. The only risk when we use this model is the possibility of generating infeasible instances because the model is not strictly constrained to fulfil all trips; but in that case, this drawback could be handled by using the software as a"what-if" tool for tuning good solutions.

The software tool was developed to provide a friendly environment for user interaction. Some alternatives to regular (weekly) schedules were also included within the tool, for example, it permits the user to predefine the rest days in advance, by fixing some specific variables of the model. The complexity of the problem when its size increases leads to high execution times, which is faced by implementing a heuristic algorithm that combines warm start strategies with an iterative local search routine.

Practical results are very encouraging, showing a strong reduction in the execution time for medium and large-scale instances, reaching the optimal solution or stopping very close to it. The computerized tool is currently operating in the train transportation company, but the model could be adapted to tackle other kinds of crew scheduling problems, especially those arising in urban and interurban buses. We have focused on this specific train industry rather than on generic transportation essentially due to the

fact that it is difficult to do shortcuts on the tree-like network, thus requiring a round trip for some destinations. Moreover, in this real case we can suppose full availability of material and human resources, and the model for the scarcity case would be out of scope, falling in the context of stochastic or robust optimization.

The main benefits for the company are not only economic in nature (in terms of person-hours dedicated to these tasks, which fell from weeks to hours), but also the tool has facilitated negotiations between management and the unions, because the system provides objective and imaginative solutions to aid the decision-making process. Moreover, the speed of calculations permits the user to generate in some minutes several optimal or quasi-optimal diagrams (in terms of balanced loads for the crews) which must be compared with the traditional process based on exchanging hand-made files during 2 or 3 weeks.

Acknowledgements The authors thank FCAB-Antofagasta Minerals Co. for support and real data testing and Andrew Hart for his careful reading and criticism that improved the quality of the article. Reviewers are also acknowledged for helpful comments and suggestions.

References

Abbink E, Wout JV, Huisman D (2007) Solving large scale crew scheduling problems by using iterative partitioning. In: Liebchen C, Ahuja RK, Mesa JA (eds), 7th Workshop on Algorithmic approaches for transportation modeling, optimization, and systems (ATMOS'07), volume 7 of OpenAccess Series in Informatics (OASIcs) (Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik)

Beasley J, Cao B (1996) A tree search algorithm for the crew scheduling problem. Eur J Oper Res 94(3):517–526

Beliën J, Demeulemeester E, Bruecker PD, den Bergh JV, Cardoen B (2013) Integrated staffing and scheduling for an aircraft line maintenance problem. Comput Oper Res 40(4):1023–1033

Desrochers M, Soumis F (1989) A column generation approach to the urban transit crew scheduling problem. Transp Sci 23(1):1–13

Emden-Weinert T, Proksch M (1999) Best practice simulated annealing for the airline crew scheduling problem. J Heuristic 5(4):419–436

Ernst A, Jiang H, Krishnamoorthy M, Nott H, Sier D (2001) An integrated optimization model for train crew management. Ann Oper Res 108(1):211–224

Ernst A, Jiang H, Krishnamoorthy M, Sier D (2004) Staff scheduling and rostering: a review of applications, methods and models. Eur J Oper Res 153(1):3–27

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

Jian MS, Chou TY (2010) Multiobjective genetic algorithm to solve the train crew scheduling problem. In: Proceedings of the 10th WSEAS international conference on systems theory and scientific computation, ISTASC'10 (Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS)), pp 100–106

Jütte S, Albers M, Thonemann UW, Hasse K (2011) Optimizing railway crew scheduling at DB Schenker. Interfaces 41(2):109–122

Levine D (1996) Application of a hybrid genetic algorithm to airline crew scheduling. Comput Oper Res 23(6):547–558

Karsu Ö, Morton A (2015) Inequity averse optimization in operational research. Eur J Oper Res 245(2):343–359

Mesquita M, Moz M, Paias A, Pato M (2015) A decompose-and-fix heuristic based on multi-commodity flow models for driver rostering with days-off pattern. Eur J Oper Res 245(2):423–437

Nishi T, Sugiyama T, Inuiguchi M (2014) Two-level decomposition algorithm for crew rostering problems with fair working condition. Eur J Oper Res 237(2):465–473 ISSN 0377-2217

Şahin G, Yüceoğlu B (2011) Tactical crew planning in railways. Transp Res Part E: Log Transp Rev 47(6):1221–1243

Sanders P, Takkula T, Wedelin D (1999) High-performance computing and networking. In: 7th International conference, HPCN Europe 1999 Amsterdam, April 12–14, 1999 Proceedings, chapter High performance integer optimization for crew scheduling, Springer, Berlin, pp 1–12

Vaidyanathan B, Jha KC, Ahuja RK (2007) Multicommodity network flow approach to the railroad crew-scheduling problem. IBM J Res Dev 51(3):325–344

Walker CG, Snowdon JN, Ryan DM (2005) Simultaneous disruption recovery of a train timetable and crew roster in real time. Comput Oper Res 32(8):2077–2094

Yunes TH, Moura AV, de Souza CC (2005) Hybrid column generation approaches for urban transit crew management problems. Transp Sci 39(2):273–288