Les Cahiers du GERAD ISSN: 0711–2440

Integrated and sequential solution methods for the cyclic bus driver rostering problem

S. Er-Rbib, A. Bani,
G. Desaulniers, I. El Hallaoui

G–2019–11

February 2019

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2019
– Bibliothèque et Archives Canada, 2019

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2019
– Library and Archives Canada, 2019

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Integrated and sequential solution methods for the cyclic bus driver rostering problem

Safae Er-Rbib
Abderrahman Bani
Guy Desaulniers
Issmail El Hallaoui

GERAD & Department of Mathematics and Industrial Engineering, Polytechnique Montréal (Québec) Canada, H3C 3A7

safae.er-rbib@gerad.ca
Abderrahman.bani@gerad.ca
guy.desaulniers@gerad.ca
issmail.elhallaoui@gerad.ca

February 2019
Les Cahiers du GERAD
G–2019–11

Abstract: Given a set of duties to be operated over a cyclic one-week horizon and groups of drivers with similar characteristics, the cyclic bus driver rostering problem (CBDRP) consists of building driver schedules that cover every duty exactly once, can cycle from one week to the next in the same driver group, and satisfy a set of labor law and collective agreement rules. The objective aims at balancing as much as possible the workload between the drivers. In this paper, we first propose an integrated mixed-integer linear programming model for the CBDRP that assigns simultaneously days off and duties to bus drivers. This model turns out to be very hard to solve to optimality without providing an initial solution. Based on this model, we introduce a new two-step matheuristic that can compute high-quality solutions. Using such a solution as an input to a commercial solver, the integrated model can be solved much more rapidly. Our computational results obtained on real-world CBDRP instances involving up to 333 drivers and 1509 duties show that these initial solutions are optimal in most cases and, consequently, that the proposed matheuristic is very efficient by itself.

Keywords: Public transit, driver rostering, cyclic rosters, days off scheduling, duty assignment, integer programming, matheuristic

Acknowledgments: We are thankful to Charles Fleurent and Marc Gendron from GIRO Inc. for introducing this problem to us, providing the data, and discussing our progress throughout the project. We gratefully acknowledge the financial support provided by GIRO Inc. and the Natural Sciences and Engineering Research Council of Canada under the grant RDCPJ 4634633–14.

1 Introduction

Public transit is a very popular way of transportation as it allows easy travel at low fares. Public transit companies are constantly required to improve the quality of their services to enhance population mobility and reduce traffic congestion. Operating costs impact the service quality, which prompts transit companies to improve their processes in order to manage as efficiently as possible their resources (buses, drivers, ...) at all planning levels: strategic, tactical and operational. Because of their size and complexity, the operations management problems faced by public transit companies are increasingly attracting operations researchers.

In this introduction, we first present an overview of the major planning steps in public transport and briefly state the problem of interest for this paper, namely, the cyclic bus driver rostering problem (CBDRP). We then review the literature on this topic and position our work with respect to this literature.

1.1 Planning process and problem overview

As described in Desaulniers and Hickman [5], the operations planning process in a public transit system is usually divided in the following three planning levels. Strategic planning involves taking long-term decisions, such as network (route) design and resource acquisition. At this level, the companies typically try to maximize quality of service (e.g., number of passengers transported) while respecting budgetary restrictions. Tactical planning is performed on a seasonal basis. It mostly aims at establishing service frequencies on the bus routes and the exact timetable such that service quality is maximized taking into account the available resources. Finally, at the operational level, the decisions concern the scheduling of the buses and the drivers as well as the planning of bus maintenance operations and overnight bus parking. These problems are solved on a monthly to a weekly basis and, sometimes, on a daily basis. Unlike the previous levels, the operational phase aims at minimizing total cost.

The three most important problems of the operational planning phase are *bus scheduling*, *duty scheduling*, and *driver rostering*. Bus scheduling consists of determining a least-cost feasible set of schedules for the available fleet of buses such that it covers all trips of a planned timetable (hereafter called active trips) over a given day. The buses may be assigned to one or several depots and may be of different types. A bus schedule is defined by a sequence of active trips to cover. It starts and ends at the same depot and may contain deadhead trips (travels without passengers) between two consecutive active trips or between the depot and an active trip.

Drivers are also assigned to depots, i.e., each driver can only drive a bus assigned to its depot. Therefore, the problem of determining the schedules of the drivers is separable by depot. For medium to large-sized instances, this problem is complex and is usually solved in two steps. In the duty scheduling step, anonymous duties are built to cover at minimum cost a set of trip segments operated out of the same depot. A *duty* represents the activities to be performed by a driver on a given day. It is defined as a sequence of (active or deadhead) trip segments, possibly interspersed with breaks and repositioning moves, that respects various feasibility rules (e.g., a maximum working time per duty and a maximum working time without a break). Given that the duties are anonymous, the problem can be separated by day. Global constraints on the maximum number of duties of each type (straight or split, long or short, early, afternoon, or late, etc.) that can be built are often considered. These constraints are derived from the set of available drivers and their preferences. The objective of the problem is to minimize the driver wages. Given that the trip timetable is defined over a week, the duties may vary from one day of the week to another but are the same from one week to another over a long planning horizon (e.g., one year) except on Holidays.

Once the duties are computed, drivers must be assigned to them. This assignment process is realized in the driver rostering step which determines at the same time the working days and the days off of each driver over a planning horizon. In the majority of the North American companies, this process is performed in order of seniority, leaving little room for optimization. In Europe, equity

among the employees often prevails and the main objective is to balance the workload between the drivers as much as possible over a week. Moreover, to increase fairness in the schedules assigned to the drivers over the planning horizon, the drivers are partitioned into groups of drivers having similar preferences with regard to their schedules (e.g., working four days per week or early duties) and who will share their weekly schedules in a cyclic manner. More specifically, for a group of n drivers, if driver i, $i = 1, \ldots, n$, is assigned to schedule s_i in a given week, then he/she is assigned to schedule s_{i+1} (or s_1 if $i = n$) in the following week. This set of schedules is called a *roster* and the resulting problem of computing simultaneously the rosters for all groups is called the CBDRP. Note that its definition depends on the labor law and collective agreement rules that each roster must respect (e.g., a minimum number of days off per week and a minimum rest time between two consecutive working days) and may, thus, differ from one bus operator to another. Additional features (such as shifts and standby duties) might also be considered to yield different CBDRP variants. In Section 2, we provide a detailed definition of the CBDRP faced by some European companies.

1.2 Literature review

Rostering or personnel scheduling problems arise in various domains of application (see Ernst et al. [6] and Van den Bergh et al. [17]): transportation, healthcare, production, retail, hospitality, finance, etc. From one domain to another, their definition may vary in different ways. Broadly speaking, these problems aim at determining the work schedule of each available employee over a planning horizon. These schedules, which must respect various operational (labor law and union) rules that differ from one domain to another, specify the days off and working days for each employee and, for each working day, the corresponding schedule and the task(s) to be accomplished. In public transit, the task assigned to a driver on a given day is a duty and, given that most duties do not have the same schedule, the drivers may be assigned to a wide variety of working schedules (essentially, a duty can start at any time during the day and last any duration in minutes between, e.g., 3 and 10 hours). In other transportation modes such as extra-urban rail and air transportation where the crew members travel long distances, crew members are not assigned to daily tasks but rather to rotations (pairings) which contain several duties that are separated by rest periods outside their home base. Furthermore, in many other sectors such as healthcare, production and call centers, employees work on a very limited number of shifts (e.g., three to five per day). In this case, the rostering problem corresponds to a generalized set covering problem (a number of employees is required for each shift), rather than a set partitioning problem (one employee per duty) like in public transit. For these reasons, we focus below on works published on public transit rostering.

In the last twenty-five years, several papers have been published on public transit rostering. Given the wide variety of contexts arising in practice, these papers have tackled different problem variants and proposed different solution approaches. Table 1 lists them and summarizes some of their characteristics. In the following, we specify the meaning of each column in this table and, if worthwhile, comment on the corresponding characteristic. The second column provides the transportation mode (bus, commuter train or subway) on which the paper focused. Most papers concern bus transportation. The next column indicates whether or not cyclic rosters are built. Non-cyclic rosters may be sought when, e.g., individual driver preferences or vacations must be taken into account or when the trip timetable is not highly regular from one week to another. In the fourth column, we indicate if days off scheduling is explicitly considered. In the three papers [4, 1, 10] where it is not, sequences of consecutive duties are computed without mentioning how these sequences are assembled to form rosters including days-off. Note also that, in [9, 11], it is assumed that a set of possible days-off patterns is provided as an input to the problem.

The fifth column specifies whether shifts (or duty types) are considered in the problem. When they are, they allow to partition the duties by shift as each duty belongs to a single shift (e.g., morning, afternoon, evening). Shifts are used to define operational rules and preferences. For instance, instead of imposing a minimum rest time between duties on two consecutive days, valid shift sequences (e.g., an evening shift can be followed by an afternoon or an evening shift only) are stipulated. Furthermore,

Table 1: Papers on rostering in public transit and some of their characteristics.

Reference	Mode	Cyclic	Days off	Shifts	Minimize	Approach	Exact	Methodology
Carraresi and Gallo [4]	bus	no	no	no	maximum workload	duty sequencing only	no	bottleneck assignment heuristic
Bianco et al. [1]	bus	no	no	no	roster costs maximum workload	duty sequencing only	no	linear programming & bottleneck assignment heuristic
Caprara et al. [3]	rail	yes	yes	no	number of drivers	integrated	no	Lagrangian relaxation & constructive heuristic
Sodhi and Norris [16]	subway	yes	yes	yes	preference-related penalties soft constraint violations	1) day-off/shift assignment 2) duty assignment	no	integer programming
Lezaun et al. [9]	bus	no	from fixed pattern set	yes	preference-related penalties shift assignment variance	1) yearly reserve scheduling 2) weekly day-off/shift assignment 3) seasonal schedule construction 4) driver assignment	no	integer programming
Moz et al. [12]	bus	no	yes	no	maximum workload number of under-utilized employees	integrated	no	evolutionary heuristics
Hartog et al. [7]	train	yes	yes	yes	preference-related penalties	1) duty set partitioning 2) day-off/shift assignment 3) duty assignment	no	integer programming
Nurmi et al. [15]	bus	no	yes	yes	soft constraint violations	1) day-off assignment 2) shift assignment	no	evolutionary heuristic
Ma et al. [10]	bus	no	no	no	workload variance	duty sequencing only	no	genetic algorithm
Nishi et al. [14]	train	yes	yes	no	maximum workload number of rosters	integrated	yes	Benders decomposition
Xie and Suhl [18]	bus	yes/no	yes	yes	maximum workload soft constraint violations preference-related penalties	integrated & 1) days-off/shift assignment 2) duty assignment	yes/no	integer programming
Borndörfer et al. [2]	bus	yes	yes	no	roster costs soft constraint violations	integrated	no	local search
Mesquita et al. [11]	bus	no	from fixed pattern set	yes	roster costs preference-related penalties	integrated	no	integer programming matheuristic

some valid sequences might be preferable to others (e.g., a morning shift followed by a morning shift is better than followed by an evening shift). In some cases, instead of requesting one driver for each shift, the demand is expressed as a number of required drivers by shift, greatly reducing the combinatorial difficulty of the problem.

The next column is devoted to the objective function considered. As can be observed, multiple objectives, which are combined into a single one using weights, are often taken into account. A frequent objective is to balance the workload between the drivers as much as possible, either by minimizing the maximum workload or the workload variance. Other objectives includes minimizing the roster costs which essentially boils down to minimizing the number of drivers, minimizing soft constraint violations, and minimizing preference-related penalties (e.g., to avoid assigning a morning shift followed by an evening shift). f The general solution approach selected to tackle the problem is provided in the seventh column. Five papers proposed an integrated approach, four a sequential one and Xie and Suhl [18] developed both approaches. The remaining three papers [4, 1, 10] focused on duty sequencing only. The eighth column indicates whether the overall solution algorithm is exact or not. Note that, to have an exact algorithm, an integrated approach must have been proposed. One can observe that only two exact algorithms were developed by Nishi et al. [14] and Xie and Suhl [18]. Finally, the last column stipulates the types of methodology used, namely, mathematical programming tools and various types of heuristics.

Among the papers listed in Table 1, those of Nishi et al. [14] and Xie and Suhl [18] are the most interesting from our point of view because we aim at solving large-sized CBDRP instances (i.e., with more than 1000 duties) exactly. Nishi et al. [14] developed a Benders decomposition algorithm that was only able to solve small-sized instances (with up to 70 duties) within a 1-hour time limit. On the other hand, Xie and Suhl [18] considered a cyclic and a non-cyclic rostering problem, proposed multi-commodity network flow formulations for them and designed an integrated approach and a two-step approach (day-off/shift assignment first, duty assignment second) that can solve large problem instances to optimality. In particular, they tested their integrated approach on a single cyclic problem instance involving 9 rosters, 11 shifts, 1013 duties and 303 standby duties, which was solved in 91 minutes. It should be noted, however, that around 12% of the activities are preassigned for this instance and that preferences for certain shifts and certain shift sequences restrict the solution space. Interestingly, the authors showed that with the integrated approach, they were able to significantly improve the quality of the solutions produced by the sequential approach.

1.3 Contributions and paper structure

The research project conducted for this paper has been defined in collaboration with the company Giro Inc., a world-leader in the development and commercialization of optimization softwares for public transport companies with more than 200 customers worldwide. The goal is to devise an exact algorithm for solving one of the most difficult CBDRPs faced by some of Giro's customers. In this problem, there are no shifts and, therefore, no shift-related rules that reduce the solution space, nor shift-related preferences that can reduce symmetry in the branch-and-bound process and ease the search for integer solutions. Furthermore, all duties can be assigned to most rosters if not all of them. Under these conditions, the network underlying the model of Xie and Suhl [18] would contain (for an instance of similar size) much more arcs than the one they tested and the computational time would get much larger as the authors acknowledged in their paper. Consequently, our main goal is to design an integrated solution approach for solving the CBDRP in this context.

In the context of the CBDRP without shifts, the main contributions of this paper are:

- We introduce a mixed-integer linear programming (MILP) model that can be solved using a commercial MILP solver. With this model, medium-sized CBDRP instances with up to 146 drivers and 640 duties are solved to optimality within two hours of computational time.

- Based on this model, we devise a two-step matheuristic that can produce high-quality solutions (optimal ones, most of the times) in relatively fast computational times for large-sized CBDRP instances.
- We show that warm-starting the integrated approach with the solution computed by the sequential approach allows to significantly reduce the computational times of the MILP solver and solve to optimality large-sized CBDRP instances.

Our computational results also show that the proposed approaches can produce better quality solutions than the solution algorithm currently used by our industrial partner Giro.

This paper is organized as follows. First, the CBDRP considered in this paper is stated in details in Section 2. Then, the integrated MILP model for this problem is introduced in Section 3, before describing the proposed solution algorithms in Section 4. Next, the computational results obtained on real-world instances provided by Giro are presented and discussed in Section 5. Finally, conclusions are drawn in Section 6.

2 Problem statement

The CBDRP is defined over a one-week horizon $\mathbf{J}$ (from Monday to Sunday) which is assumed to repeat for a long period of time (e.g., a season or one year). For each day $j \in \mathbf{J}$, a set of (regular) duties $\mathbf{D}_j^R$ must be operated. Each duty $d \in \mathbf{D}_j^R$, $j \in \mathbf{J}$, is defined by a starting time S_d, an ending time E_d, a set of bus lines served in the duty $\mathbf{L}_d$, and a workload W_d. The duty workload corresponds to the time paid to the assigned driver. Depending on the duty type, it may differ from the duty length (which is equal to $E_d - S_d$). For instance, the workload is less than the duty length for a split duty which is composed of two parts usually covering the peak hours, separated by a long break period (several hours).

The duties must be assigned to a set of available drivers which are partitioned into groups of drivers having similar preferences. For instance, all drivers preferring duties with a high workload to work only four days per week may be grouped together. The weekly schedules assigned to the drivers of a group form a roster that must be cyclic in the following sense. These schedules are assigned to a position in a cycle (from one to the number of schedules). A driver assigned to position i in a given week will be assigned to position $i+1$ (or to position 1 if he/she was in last position) in the following week. Consequently, for a roster with n positions, the cycle has a length of n weeks, which means that, after n weeks, the drivers assigned to this roster will have worked the same exact schedules but on different weeks. Note that the roster feasibility rules (stated below) apply to this cycle: for example, if a driver cannot work more than six consecutive days, then the sum of the number of work days at the end of the schedule in any position i and at the beginning of the schedule in position $i+1$ must not exceed six.

Table 2 presents an example of a roster with four positions. In this table, OFF indicates a day off while d_i, $i = 1, \ldots, 8$, represent duties assigned to the drivers on the remaining days. Notice that the same duty may appear more than once but on different days. In this example, the roster respects a maximum of six consecutive working days and the assignment of two days off per week (position).

Table 2: Example of a roster with four positions (drivers).

Position	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	d_1	d_5	OFF	OFF	d_6	d_8	d_4
2	d_2	d_6	d_1	OFF	OFF	d_3	d_5
3	d_3	d_7	d_2	d_4	OFF	OFF	d_6
4	d_4	d_8	d_3	d_5	d_7	OFF	OFF

The set of rosters $\mathbf{R}$ to construct and the set of positions $\mathbf{P}_r$ to fill in each roster $r \in \mathbf{R}$ are given as input to the CBDRP. Given the type of schedules sought for each roster, it may be forbidden to

assign some duties to a roster. Consequently, each duty $d \in \mathbf{D}_j^R$, $j \in \mathbf{J}$, is associated with a set of rosters $\mathbf{R}_d$ to which it can be assigned.

Given the complexity of the roster feasibility rules and that the number of duties may vary from one day to another, it is difficult to perfectly match the number of available drivers with the number of duties to operate each day of the week. Therefore, it often happens that there is an excess of drivers on a given day. Instead of assigning extra days off to some drivers, it is common practice to assign them to a standby duty during which they may be called to replace another driver in case of absence. Thus, for each day $j \in \mathbf{J}$, there is a set of standby duties $\mathbf{D}_j^S$ that might or might not be assigned to one or several drivers. Each standby duty $d \in \mathbf{D}_j^S$ is defined by a starting time S_d and an ending time E_d. Given that the driver assigned to such a duty might not be called in, no workload is associated with the standby duties. Note that, in general, the number of available drivers is sufficient to cover all regular duties when the groups of drivers are well designed. Hence, we enforce the covering of all these duties.

Rosters must conform to safety regulations and collective agreement rules that define hard constraints to meet. These rules may vary from one company to another. According to our industrial partner, those stated in Table 3 are the most important and common ones. The specific values indicated in this table (e.g., 13 hours in Rule 3) are those used for our computational experiments. They might slightly differ for other practical instances. Note that, for our experiments, there are two roster types: one denoted $\mathbf{R}^2$ in which two days off must be granted to each position, and another denoted $\mathbf{R}^3$ with three days off per position (thus, $\mathbf{R} = \mathbf{R}^2 \cup \mathbf{R}^3$). We consider that all these roster feasibility rules are strict, i.e., they cannot be violated.

Table 3: Roster feasibility rules.

Rule	Description
1	Depending on the roster type, there must be exactly 2 or 3 days off per position, with at least two consecutive days off in each position. Due to the cyclic nature of the rosters, consecutive days off on Sunday and Monday can serve as a double day off on either position (that with the Sunday off or that with the Monday off) but not both.
2	The number of consecutive days without a day off cannot exceed 6.
3	There must be at least 9 hours of rest between two consecutive duties.
4	There must be at least 10 hours of rest between two consecutive duties if they are immediately followed by a third duty.
5	A 57-hour period of rest, including at least two consecutive days off, must be assigned to each position.
6	In any period of 28 consecutive days ending with a working day, the average rest time between the duties must be at least 12 hours.
7	At most one duty lasting 13 hours or more can be assigned to each position.
8	At most two duties serving the same bus line can be assigned consecutively.

The CBDRP can be stated as follows. Given the sets of regular and standby duties $\mathbf{D}_j^R$ and $\mathbf{D}_j^S$ for $j \in \mathbf{J}$, and the set of rosters $\mathbf{R}$, assign duties and days off to the rosters such that all regular duties are assigned to exactly one position on the days they must be operated, standby duties are assigned only if needed, all rosters are feasible with respect to Rules 1 to 8, and the total workload is balanced as much as possible between all positions. This balancing can be formulated in different ways. For instance, if $A = \frac{\sum_{j \in \mathbf{J}} \sum_{d \in \mathbf{D}_j^R} W_d}{\sum_{r \in \mathbf{R}} |P_r|}$ denotes the average workload per position, then one can minimize the sum over all positions of the absolute deviations from A or the squares of these deviations. In our case, our objective function seeks to minimize the sum of the positive deviations, i.e., only over the positions for which the workload exceeds A.

3 A mathematical model for the CBDRP

In this section, we present a mathematical model for the CBDRP defined in the previous section and discuss some of its characteristics. Beforehand, we introduce the necessary notation.

3.1 Notation

For the sake of completeness, we re-introduce the notation presented above. Note that this notation respects the following convention: sets are denoted using bold capital letters, variables using lowercase letters, and parameters using capital letters.

3.1.1 Sets

$\mathbf{J}$: Set of days in a week, numbered from 0 (Monday) to 6 (Sunday);
$\mathbf{D}_j^R$: Set of duties that must be assigned on day j;
$\mathbf{D}_j^S$: Set of standby duties that can be assigned (once or more) on day j;
$\mathbf{R}$: set of all rosters;
$\mathbf{R}^i$, $i \in \{2, 3\}$: Set of rosters with i days off per position;
$\mathbf{R}_d$: Set of rosters to which duty d can be assigned;
$\mathbf{P}$: Set of all positions in all rosters;
$\mathbf{P}_r$: Set of positions in roster r, numbered from 0 to $|\mathbf{P}_r| - 1$;
$\mathbf{D}_{pj}^R$: Set of regular duties that can be assigned to position p on day j;
$\mathbf{D}_{pj} = \mathbf{D}_{pj}^R \cup \mathbf{D}_j^S$: Set of regular and standby duties that can be assigned to position p on day j;
$\mathbf{L}$: Set of bus lines;
$\mathbf{L}_d$: Set of bus lines served in duty d;

3.1.2 Parameters

W_d : Workload in duty d (in minutes);
S_d : Starting time of duty d (in minutes from the beginning of the week);
E_d : Ending time of duty d (in minutes from the beginning of the week);
I_d^{13} : Binary indicator equal to 1 if the workload of duty d exceeds 13 hours;
I_{dl} : Binary indicator equal to 1 if duty d serves bus line l, i.e., if $l \in L_d$;
A : Average workload per position ($A = \frac{\sum_{j \in \mathbf{J}} \sum_{d \in \mathbf{D}_j} W_d}{\sum_{r \in \mathbf{R}} |P_r|}$);
M : A sufficiently large constant.

3.1.3 Variables

x_{pj}^d : Binary variable equal to 1 if duty d is assigned to position p on day j, and to 0 otherwise;
y_{pj}^2 : Binary variable equal to 1 if a double day off is assigned to position p starting on day j, and to 0 otherwise;
y_{pj}^1 : Binary variable equal to 1 if a single day off is assigned to position p on day j, and to 0 otherwise;
z_p : Nonnegative variable equal to the excess workload in position p with respect to the average workload per position, i.e., $z_p = \max\{0, \sum_{j \in \mathbf{J}} \sum_{d \in \mathbf{D}_{pj}^R} W_d x_{pj}^d - A\}$;
u_{pj} : Binary variable equal to 1 if the driver in position p is assigned to a duty on day j and on the following day, and to 0 otherwise;
v_{pj} : Nonnegative variable equal to the duration of the rest period between day j and the following day if the driver in position p is assigned to a duty on day j and on the following day, and to 0 otherwise.

3.1.4 Functions

$r(p)$: Roster of position p.

A roster r can be seen as the following cycle of pairs (p, j) of position p and day j: $(0,0) - (0,1) - \ldots - (0,6) - (1,0) - (1,1) - \ldots - (|P_r|-1, 5) - (|P_r|-1, 6) - (0,0)$. In the proposed model, we often need

to identify which pair (p^*, j^*) is reached when moving k days forward (if $k > 0$) or backward (if $k < 0$) from a pair (p, j) in the cycle. For instance, for a roster r with $|P_r| = 4$ positions, $(p^*, j^*) = (3, 6)$ if $(p, j) = (4, 0)$ and $k = -1$ and $(p^*, j^*) = (0, 2)$ if $(p, j) = (4, 5)$ and $k = 4$. In this regard, we introduce a vectorial function[1]

$$f(p, j, k) = \left(\lfloor p + \frac{(j+k)}{7} \rfloor \bmod |P_{r(p)}| \, , \, (j+k) \bmod 7 \right) \tag{1}$$

which returns this position-day pair (p^*, j^*) for a given triplet of position p, day j and number of days shifted k. With this function, writing $x^d_{f(4,5,4)}$ is equivalent to writing $x^d_{0,2}$ if $|P_{r(4)}| = 4$ as in the last example above.

3.2 Model

Using the notation presented above, the CBDRP can be modeled as the following mixed-integer linear program, which is called the *integrated model* hereafter:

$$\min \quad \sum_{p \in \mathbf{P}} z_p \tag{2}$$

subject to

$$z_p \geq \sum_{j \in \mathbf{J}} \sum_{d \in \mathbf{D}^R_{pj}} W_d x^d_{pj} - A, \qquad \forall p \in \mathbf{P} \tag{3}$$

$$z_p \geq 0, \qquad \forall p \in \mathbf{P} \tag{4}$$

$$\sum_{r \in \mathbf{R}_d} \sum_{p \in \mathbf{P}_r} x^d_{pj} = 1, \qquad \forall j \in \mathbf{J}, d \in \mathbf{D}^R_j \tag{5}$$

$$\sum_{d \in \mathbf{D}_{pj}} x^d_{pj} + y^1_{pj} + y^2_{pj} + y^2_{f(p,j,-1)} = 1, \qquad \forall p \in \mathbf{P}, j \in \mathbf{J} \tag{6}$$

$$\sum_{j \in \mathbf{J}} y^2_{pj} = 1, \qquad \forall p \in \mathbf{P} \tag{7}$$

$$\sum_{j \in \mathbf{J}} y^1_{pj} = 1, \qquad \forall r \in \mathbf{R}^3, p \in \mathbf{P}_r \tag{8}$$

$$y^2_{f(p,j,-1)} + \sum_{k=0}^{6} \left(y^2_{f(p,j,k)} + y^1_{f(p,j,k)} \right) \geq 1, \qquad \forall p \in \mathbf{P}, j \in \mathbf{J} \tag{9}$$

$$2u_{pj} \leq \sum_{d \in \mathbf{D}_{pj}} x^d_{pj} + \sum_{d \in \mathbf{D}_{f(p,j,1)}} x^d_{f(p,j,1)} \leq 1 + u_{pj}, \qquad \forall p \in \mathbf{P}, j \in \mathbf{J} \tag{10}$$

$$v_{pj} \leq M \sum_{d \in \mathbf{D}_{pj}} x^d_{pj}, \qquad \forall p \in \mathbf{P}, j \in \mathbf{J} \tag{11}$$

$$v_{pj} \leq M \sum_{d \in \mathbf{D}_{f(p,j,1)}} x^d_{f(p,j,1)}, \qquad \forall p \in \mathbf{P}, j \in \mathbf{J} \tag{12}$$

$$-M(1 - u_{pj}) \leq v_{pj} - \Big(\sum_{d \in \mathbf{D}_{f(p,j,1)}} S_d x^d_{f(p,j,1)} - \sum_{d \in \mathbf{D}_{pj}} E_d x^d_{pj} \Big) \leq M(1 - u_{pj}), \qquad \forall p \in \mathbf{P}, j \in \mathbf{J} \tag{13}$$

$$v_{pj} \geq 540 u_{pj}, \qquad \forall p \in \mathbf{P}, j \in \mathbf{J} \tag{14}$$

$$v_{pj} \geq 600 u_{pj} - M\Big(1 - \sum_{d \in \mathbf{D}_{f(p,j,2)}} x^d_{f(p,j,2)}\Big), \qquad \forall p \in \mathbf{P}, j \in \mathbf{J} \tag{15}$$

[1] To always return a nonnegative value, we define the modulo function as follows: $a \mod n = a - n \lfloor \frac{a}{n} \rfloor$.

$$\sum_{d\in\mathbf{D}_{f(p,j,3)}} S_d x^d_{f(p,j,3)} - \sum_{d\in\mathbf{D}_{pj}} E_d x^d_{pj} \geq 3420 - M(3 - y^2_{f(p,j,1)} - \sum_{d\in\mathbf{D}_{pj}} x^d_{pj} - \sum_{d\in\mathbf{D}_{f(p,j,3)}} x^d_{f(p,j,3)}), \quad \forall p \in \mathbf{P}, j \in \mathbf{J} \tag{16}$$

$$\sum_{k=0}^{27} v_{f(p,j,k)} \geq 720 \sum_{k=0}^{27} u_{f(p,j,k)} - M(1 - \sum_{d\in\mathbf{D}_{f(p,j,27)}} x^d_{f(p,j,27)}), \quad \forall p \in \mathbf{P}, j \in \mathbf{J} \tag{17}$$

$$\sum_{j\in\mathbf{J}} \sum_{d\in\mathbf{D}_{pj}} I^{13}_d x^d_{pj} \leq 1, \quad \forall p \in \mathbf{P} \tag{18}$$

$$\sum_{d\in\mathbf{D}_{pj}} I_{dl} x^d_{pj} + \sum_{d\in\mathbf{D}_{f(p,j,1)}} I_{dl} x^d_{f(p,j,1)} + \sum_{d\in\mathbf{D}_{f(p,j,2)}} I_{dl} x^d_{f(p,j,2)} \leq 2, \quad \forall p \in \mathbf{P}, j \in \mathbf{J}, l \in \mathbf{L} \tag{19}$$

$$x^d_{pj} \in \{0,1\}, \quad \forall p \in \mathbf{P}, j \in \mathbf{J}, d \in \mathbf{D}_{pj} \tag{20}$$

$$y^2_{pj}, y^1_{pj}, u_{pj} \in \{0,1\}, \quad \forall p \in \mathbf{P}, j \in \mathbf{J}. \tag{21}$$

Combined with constraints (3) and (4), the objective function (2) minimizes the sum over all positions of the workload exceeding the average workload per position in order to balance as much as possible the workload among the positions. Constraints (5) ensure that every regular duty is assigned to a position. Constraints (6) specify that a duty or a day off must be assigned to each position on each day. Rule 1 on the days off assignment per position is imposed through constraints (7) and (8). Constraints (9) limit the number of consecutive working days as dictated by Rule 2. In fact, it says that there must be at least one day off in every period of seven consecutive days.

Next, constraints (10) set the values of the u_{pj} variables in function of the x^d_{pj} variables, while the values of the v_{pj} variables are computed through the constraints (11)–(13). Recall that v_{pj} must be set to zero if a duty is not assigned to position-day pair (p, j) or position-day pair $f(p, j, 1)$. When not equal to zero, v_{pj} is equal to rest duration between the duties assigned on the position-day pairs $f(p, j, 1)$ and (p, j), i.e., the difference between the start time of the first of these duties and the end time of the second. As imposed through constraints (14) and (15), it must be larger than 9 hours (540 minutes) according to Rule 3 or 10 hours (600 minutes) according to Rule 4, respectively. Rule 5 on a rest period of at least 57 hours (3420 minute) per position is enforced through constraints (16). Given that there is exactly one double day off per position and that this rest must include it, it becomes possible to model this rule using one constraint per day and position. Note that, in constraints (13) and (16), one week (i.e., 10800 minutes) must be added to all parameters S_d if position p differs from the position associated with $f(p, j, 1)$ or $f(p, j, 3)$, respectively.

Rule 6 on the average rest time on every period of 28 days is imposed through constraints (17). In these constraints, the first sum computes the total rest time between two consecutive working days while the second sum computes the total number of pairs of consecutive working days. When the last of the 28 days is a working day (i.e., $\sum_{d\in\mathbf{D}_{f(p,j,27)}} x^d_{f(p,j,27)} = 1$), the ratio of these two first sums gives the average rest time over these 28 days which must exceed 12 hours (720 minutes). Rule 7 limiting the number of duties with 13 hours or more per position is imposed by constraints (18), while Rule 8 on the maximum number of consecutive days servicing the same line is enforced by constraints (19). Finally, binary restrictions on the decision variables are expressed through (20) and (21).

3.3 Model characteristics

Assuming that $|\mathbf{J}|$ does not vary ($|\mathbf{J}| = 7$), model (2)–(21) contains $O(|\mathbf{P}|^2)$ variables and $O(|\mathbf{P}||\mathbf{L}|)$ constraints because $|\mathbf{D}^R_j| \leq |\mathbf{P}|$ for all $j \in \mathbf{J}$ in any feasible instance. Thus, the size of this model grows rapidly with the number of positions. In fact, as it will be reported in Section 5, the number of variables counts in hundreds of thousands of variables for instances with a few hundreds positions.

The integrated model (2)–(21) involves many big-M constraints, namely, constraints (11)–(13) and (15)–(17). Such constraints typically yield relatively loose linear relaxations and a large number of fractional-valued variables in their optimal solutions. Therefore, when solving them by a MILP solver, large branch-and-bound search trees should be expected. Note that these big-M constraints could be avoided by replacing each of them with a relatively large number of constraints as it will be proposed in Section 4.1.2, but given the already large size of the integrated model, we prefer to keep them.

One last characteristic of this model is that it contains a large number of symmetric solutions which can significantly increase the number of nodes in the search tree. Indeed, given that the weekly schedules are anonymous, it is always possible to shift all schedules assigned to a roster by one position to obtain an equivalent solution. Furthermore, many duties appear on different days of the week and, when a subset of them are assigned on different positions on different days, it may be possible to permute them without violating any rest time constraints. Also, there often exist several duties that can be assigned to the same roster on the same day that have the same workload but maybe slightly different starting and ending times. These duties can often be permuted from one position to another without violating the rest time constraints. Finally, when such duties can be assigned to different duties, it increases the number of symmetric solutions.

To illustrate some of these symmetry cases, Tables 4 and 5 present an example with one roster, four positions and four duties that must be assigned overall on sixteen days. We assume that three days off must be assigned to each position and that all presented solutions satisfy the roster feasibility rules. The duties and their characteristics are given in Table 4. For each duty d, it provides its starting time S_d, its ending time E_d, its workload W_d and its set of days $\mathbf{J}_d$ on which it must be assigned. For convenience, the starting and ending times are given in the format $hh:mm$ and the workload in hours. Table 5 presents three symmetric solutions with the same cost. The second solution is obtained from the first by shifting the weekly schedules forward by one position, i.e., the schedule in position i, $i = 0, 1, 2$ is moved to position $i + 1$ and that of position 3 is moved to position 0. The third solution is obtained from the second by exchanging the duties d_1 and d_2 assigned to positions 1 and 3 on Wednesday and Thursday.

Table 4: The four duties and their characteristics.

Duty	S_d	E_d	W_d	$\mathbf{J}_d$
d_1	7:00	19:00	$9h$	Mon, Wed, Thu
d_2	6:00	18:00	$10h$	Tue, Wed, Thu, Sun
d_3	7:00	15:00	$7h$	Thu, Fri, Sat, Sun
d_4	9:00	18:00	$8h$	Mon, Tue, Fri, Sat, Sun

To break a part of the symmetry, one can impose for each roster r the following constraints:

$$\sum_{j \in \mathbf{J}} \sum_{d \in \mathbf{D}^R_{0j}} W_d x^d_{0j} \geq \sum_{j \in \mathbf{J}} \sum_{d \in \mathbf{D}^R_{pj}} W_d x^d_{pj}, \qquad \forall p \in P_r \setminus \{0\}, \tag{22}$$

which ensure that the first position in roster r has the maximum workload among the positions in this roster. Our computational experiments showed, however, that these constraints have little impact on the computational times.

Clearly, these characteristics of the integrated model make it difficult to solve to optimality, especially for large CBDRP instances. In the following section, we introduce, among others, a matheuristic that will turn out to also be useful for solving the problem to optimality.

4 Solution algorithms

In this section, we describe three solution algorithms for the CBDRP. The first algorithm is heuristic while the second and third ones are exact. In fact, the last algorithm combines the first two.

Table 5: Three symmetric solutions.

Solution 1							
Position	Mon	Tue	Wed	Thu	Fri	Sat	Sun
0	OFF	OFF	d_1	d_2	d_3	d_4	OFF
1	d_1	d_2	OFF	OFF	OFF	d_3	d_4
2	OFF	d_4	d_2	d_1	OFF	OFF	d_3
3	d_4	OFF	OFF	d_3	d_4	OFF	d_2
Solution 2							
Position	Mon	Tue	Wed	Thu	Fri	Sat	Sun
0	d_4	OFF	OFF	d_3	d_4	OFF	d_2
1	OFF	OFF	d_1	d_2	d_3	d_4	OFF
2	d_1	d_2	OFF	OFF	OFF	d_3	d_4
3	OFF	d_4	d_2	d_1	OFF	OFF	d_3
Solution 3							
Position	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	d_4	OFF	OFF	d_3	d_4	OFF	d_2
2	OFF	OFF	d_2	d_1	d_3	d_4	OFF
3	d_1	d_2	OFF	OFF	OFF	d_3	d_4
4	OFF	d_4	d_1	d_2	OFF	OFF	d_3

4.1 A two-step matheuristic

We propose a two-step matheuristic that schedules the days off before assigning the duties to the working days. This approach differs from the sequential ones listed in our literature review (see Table 1) by considering duty assignment variables in the days-off scheduling step. This allows to position the days off in the rosters while taking into account the workload balancing between the drivers. Let us describe each step of this algorithm which we denote TSM for two-step matheuristic.

4.1.1 Step 1: Days-off scheduling

In the days-off scheduling step, we consider the relaxation of the integrated model (2)–(21) obtained by dropping all the constraints related to Rules 3 to 8. Thus, the resulting model, called the *DOS model* hereafter for days-off scheduling, is given by (2)–(9), (20) and (21) but without the binary requirements on the u_{ij} variables that are not needed here. Note that the v_{ij} variables are neither necessary and that all constraints involving a big M constant in the complete model have been omitted. The DOS model is solved using a commercial MILP solver. From the computed solution, we retain only the scheduled days off as there is no guarantee that the duty assignment satisfies all relaxed constraints.

One can expect that the DOS model is much easier to solve than the complete model (2)–(21) for the following reasons. First, it involves less variables and much less constraints. In fact, the number of constraints is drastically reduced by about 90% as it will be shown in Section 5. Second, all constraints with a big-M constant have been removed. Such constraints are known to yield looser linear relaxations. Finally, given the much smaller number of constraints, the number of fractional-valued variables in the linear relaxation solution at the root node of the search tree should be substantially reduced. Typically, less fractional-valued variables induce less cuts to add and less branch-and-bound nodes to explore. In this case, the large number of symmetric solutions should have less impact on the solution process.

4.1.2 Step 2: Duty assignment

Given the days off computed in the first step, the duty assignment step consists of assigning duties to the working days in the rosters while taking into account only the constraints that depend on the duty assignment. Therefore, Rules 1 and 2 are not relevant. This duty assignment problem can be formulated with the integrated model (2)–(21) after fixing all y variables to the values they take in

the computed solution to the DOS model and removing the unnecessary constraints. Here we propose another model that exploits the knowledge of the days off and avoids all big-M constraints.

In addition to the previous notation, we need the following one.

Sets

$\mathbf{J}_p^W$: Set of working days in position p;

$\mathbf{J}_p^9$: Set of working days in position p which are immediately followed by a working day and a day off (Rule 3 must be checked);

$\mathbf{J}_p^{10}$: Set of working days in position p which are immediately followed by at least two consecutive days (Rules 4 and 8 must be checked on those days);

$\mathbf{J}_p^{57}$: Set containing the single working day in position p which is followed by a double day off (Rule 5 must be checked);

$\mathbf{J}_p^{12}$: Set of days in position p which start a sequence of 28 consecutive days ending with a working day (Rule 6 must be checked);

$\mathbf{K}_{pj}$: Subset of $\{0, 1, \ldots, 27\}$ indicating the pairs of consecutive working days in a sequence of 28 consecutive days starting in position-day pair (p, j), i.e., $k \in \mathbf{K}_{pj}$ if working days are assigned to both position-day pairs $f(p, j, k)$ and $f(p, j, k + 1)$.

$\mathbf{T}_{pj}$: Set of all ending times of the duties in $\mathbf{D}_{pj}$.

Parameters

F_{dt}^+ : Binary indicator equal to 1 if duty d finishes at time t or after;

B_{dt}^- : Binary indicator equal to 1 if duty d begins before time t (excluding time t);

U_{pj} : Total number of working days immediately followed by a working day in a sequence of 28 consecutive days starting in position-day pair (p, j).

With this notation, we model the duty assignment problem as the following mixed-integer linear program, called the *DA model* hereafter:

$$\min \quad \sum_{p \in \mathbf{P}} z_p \tag{23}$$

subject to

$$z_p \geq \sum_{j \in \mathbf{J}} \sum_{d \in \mathbf{D}_{pj}^R} W_d x_{pj}^d - A, \qquad \forall p \in \mathbf{P} \tag{24}$$

$$z_p \geq 0, \qquad \forall p \in \mathbf{P} \tag{25}$$

$$\sum_{r \in \mathbf{R}_d} \sum_{p \in \mathbf{P}_r} x_{pj}^d = 1, \qquad \forall j \in \mathbf{J}, d \in \mathbf{D}_j^R \tag{26}$$

$$\sum_{d \in \mathbf{D}_{pj}} x_{pj}^d = 1, \qquad \forall p \in \mathbf{P}, j \in \mathbf{J}_p^W \tag{27}$$

$$\sum_{d \in \mathbf{D}_{pj}} F_{dt}^+ x_{pj}^d + \sum_{d \in \mathbf{D}_{f(p,j,1)}} B_{d,t+540}^- x_{f(p,j,1)}^d \leq 1, \qquad \forall p \in \mathbf{P}, j \in \mathbf{J}_p^9, t \in \mathbf{T}_{pj} \tag{28}$$

$$\sum_{d \in \mathbf{D}_{pj}} F_{dt}^+ x_{pj}^d + \sum_{d \in \mathbf{D}_{f(p,j,1)}} B_{d,t+600}^- x_{f(p,j,1)}^d \leq 1, \qquad \forall p \in \mathbf{P}, j \in \mathbf{J}_p^{10}, t \in \mathbf{T}_{pj} \tag{29}$$

$$\sum_{d \in \mathbf{D}_{pj}} F_{dt}^+ x_{pj}^d + \sum_{d \in \mathbf{D}_{f(p,j,3)}} B_{d,t+3420}^- x_{f(p,j,3)}^d \leq 1, \qquad \forall p \in \mathbf{P}, j \in \mathbf{J}_p^{57}, t \in \mathbf{T}_{pj} \tag{30}$$

$$\sum_{k\in\mathbf{K}_{pj}}\Big(\sum_{d\in\mathbf{D}_{f(p,j,k+1)}} S_d x^d_{f(p,j,k+1)} - \sum_{d\in\mathbf{D}_{f(p,j,k)}} E_d x^d_{f(p,j,k)}\Big) \geq 720U_{pj}, \quad \forall p\in\mathbf{P}, j\in\mathbf{J}^{12}_p \tag{31}$$

$$\sum_{j\in\mathbf{J}^W_p}\sum_{d\in\mathbf{D}_{pj}} I^{13}_d x^d_{pj} \leq 1, \quad \forall p\in\mathbf{P} \tag{32}$$

$$\sum_{d\in\mathbf{D}_{pj}} I_{dl}x^d_{pj} + \sum_{d\in\mathbf{D}_{f(p,j,1)}} I_{dl}x^d_{f(p,j,1)} + \sum_{d\in\mathbf{D}_{f(p,j,2)}} I_{dl}x^d_{f(p,j,2)} \leq 2, \quad \forall p\in\mathbf{P}, j\in\mathbf{J}^{10}_p, l\in\mathbf{L} \tag{33}$$

$$x^d_{pj}\in\{0,1\}, \quad \forall p\in\mathbf{P}, j\in\mathbf{J}^W_p, d\in\mathbf{D}_{pj}. \tag{34}$$

Objective function (23), constraints (24)–(27) and (32)–(34) correspond to (2)–(6) and (18)–(20) in the integrated model, except that constraints (27) and (33) are restricted to the days where they apply. Constraints (28)–(30) impose Rules 3, 4 and 5, respectively. They are all based on the same rationale and, therefore, we explain only the first constraint set. For a given possible duty ending time t at position-day pair (p, j), a constraint (28) stipulates that it is not possible to assign more than one duty to position-day pairs (p, j) and $f(p, j, 1)$ that overlaps with time interval $[t, t + 540[$ without violating Rule 3 (at least 9 hours of rest between two consecutive duties). In fact, this constraint is lifted by also including all variables associated with the duties that can be assigned to pair (p, j) and start at time $t + 540$ or after. Notice here that 10800 minutes must subtracted from $t + 540$ (or $t + 600$ or $t + 3420$) when $t + 540 > 10800$. Finally, constraints (31) ensure that Rule 6 is satisfied. The left-hand side computes the total rest time in the considered sequence of 28 days which must be greater than 12 hours multiplied by the number of rests between two consecutive duties in this sequence. As for the integrated model, one week (i.e., 10800 minutes) must be added to all parameters S_d if the position of pair $f(p, j, k)$ differs from the position of pair $f(p, j, k + 1)$.

The DA model (23)–(34) is solved using a commercial MILP solver. In general, it is much easier to solve than the integrated because it has a reduced number of variables, no big-M constraints, and prone to less symmetry because of the fixed days off. Combining the computed solution of the DA model with the days off schedule obtained in the first step of the TSM yields a complete feasible solution to the CBDRP.

Observe that, in the second step of the TSM, the duty assignment problem may be infeasible even if the CBDRP is feasible. Indeed, because the fixed days off schedule was computed while relaxing some constraints to the CBDRP, there is no guarantee that a feasible duty assignment can be found with this schedule. However, given the flexibility provided by the large number of duties that can be assigned to each roster and vice versa, this situation did not occur in our computational tests. It should also be noted that this flexibility helps finding optimal or near-optimal solutions with the TSM.

4.2 Two exact algorithms

We propose two exact algorithms for the CBDRP. The first one simply consists of solving the integrated model (2)–(21) using a commercial MILP solver. This algorithm is denoted INT for integrated-model-based algorithm. It is well-known that state-of-the-art MILP solvers have several components which can help to speed up the overall solution process. One of them is a set of heuristics that searches for a good feasible solution and that is invoked at various stages of the solution process and, in particular, just after solving the first linear relaxation. A feasible solution provides an upper bound on the optimal value which can be used to fix the values of some variables according to their reduced costs (see, e.g., Nemhauser and Wolsey [13], p. 389). A better upper bound yields a larger number of variables that can be fixed at one of their bounds, typically at 0 in our case. Fixing a large number of variables may have several positive impacts on the solution process, including further preprocessing to fix other variables and remove redundant constraints, faster linear relaxation computational times incurred by the problem size reduction, increased opportunity to find a better solution around this initial solution when applying the solver heuristics, and tighter subsequent linear relaxations. As our computational

experiments will show in the next section, the heuristic embedded in the state-of-the-art MILP solver that we used (the CPLEX MILP solver) was unable to find good-quality feasible solutions and to solve the largest instance that we tested.

As a second exact algorithm, we propose to combine the TSM and INT algorithms. The TSM is first executed to compute a high-quality feasible solution which is then inputted as an initial solution to the INT algorithm. The INT algorithm can then proceed to variable fixing early in the solution process and benefit from the ensuing positive impacts. This algorithm is denoted INTIS in the following for integrated-model-based algorithm with an initial solution.

5 Computational results

To compare the performance of the three proposed algorithms, we conducted computational experiments on two real-world instances and on five additional instances derived from them. In this section, we first describe these instances. Then, we report and discuss the computational results obtained from these experiments. Finally, we compare our results with those produced by the current solution approach used by our industrial partner.

All proposed models were implemented using the IBM concert technology and solved using the CPLEX MILP solver version 12.7.0.0 with default parameter values. The computational tests were conducted on a Linux machine with an Intel Xeon E5-2637 V2 processor, clocked at 3.5 GHz and a RAM of 128 GB.

5.1 Instances

Our industrial partner provided two real-world datasets, denoted by A and G, that come from two different European public transit companies. Instance A contains 14 rosters, 202 positions, 881 duties and 14 bus lines; the rosters differ by the number of days off per position (2 or 3), by the number of positions (between 6 and 35) and by the duties they can operate though most duties can be assigned to almost all rosters. Instance G contains 2 rosters, 333 positions, 1509 duties and 28 bus lines: two day offs must be assigned to the roster with 180 drivers whereas three days off must be assigned to that with 153 drivers. For this instance, all duties can be assigned to both rosters.

From these two datasets, we randomly generated five additional smaller-sized instances as follows:

1. From dataset A, we created three instances by choosing a subset of the rosters in instance A such that these rosters contain in total between 100 and 150 positions. From dataset G which contains only two rosters, we created two instances with two rosters containing a total of 165 and 180 positions, respectively.
2. For each instance i, we computed a feasible days off schedule and, for each day of the week j, established the number n_{ij} of duties to assign according to this schedule.
3. For each instance i and each day of the week j, we selected randomly a set of n_{ij} duties that can be assigned to at least one selected roster on day j while ensuring that the resulting instance is feasible.

Table 6 presents the instances and their characteristics, namely, the dataset from which it was generated, the number of rosters, the number of positions, and the number of duties it contains. The name of an instance $I_{S,R,P}$ specifies the original dataset (S), the number of rosters (R) and the number of positions (P). Note that the last two instances $I_{A,14,202}$ and $I_{G,2,333}$ correspond to the complete datasets.

5.2 Comparative results for the INT, TSM and INTIS algorithms

We begin by presenting the results obtained by the INT algorithm, which is the most straightforward algorithm among the proposed algorithms. These results are reported in Table 7. For each instance,

Table 6: Instances and their characteristics.

Instance	Dataset	#Rosters	#Positions	#Duties
$I_{A,5,110}$	A	5	110	490
$I_{A,6,124}$	A	6	124	414
$I_{A,10,146}$	A	10	146	640
$I_{G,2,165}$	G	2	165	745
$I_{G,2,180}$	G	2	180	813
$I_{A,14,202}$	A	14	202	881
$I_{G,2,333}$	G	2	333	1509

this table indicates the numbers of variables and constraints in the integrated model, the number of branch-and-bound nodes explored (excluding the root node of the search tree) during the solution process, the number of cuts applied, the total computational time in seconds, the relative integrality gap before and after adding cuts at the root node (G^B and G^A), and the cost of the computed solution (in minutes). Note that the solution process was stopped after twelve hours of computing time (and only three branch-and-bound nodes) for the largest instance $I_{G,2,333}$, with an optimality gap exceeding 60%. All the other instances were solved within this time limit.

From these results, we observe that the initial integrality gaps (i.e., before adding cuts) are relatively small for all instances, except for instances $I_{G,2,165}$ and $I_{G,2,180}$. In all cases, the cuts are very effective at closing this gap. In fact, for the first six instances, they completely close the gap. Nevertheless, a very large number of nodes in the search tree needs to be explored for finding an optimal solution. As discussed in Section 3.3, this can be explained by the large number of symmetric solutions. Without this symmetry, the computational times would be much smaller.

Next, in Tables 8 and 9, we present the results obtained in each step of the TSM. The days off scheduling results (Table 8) show that, for all instances, the DOS model can be solved in less than 20 minutes. Remark that no branching was required and that a much smaller number of cuts were generated than for the integrated model (see Table 7). As expected, relaxing a large number of constraints from the integrated model to obtain the DOS model yields linear relaxation solutions with much less fractional-valued variables. For example, there are 4462 and 7005 fractional-valued variables in the linear relaxation solutions of the integrated model for the complete instances $I_{A,14,202}$ and $I_{G,2,333}$, respectively, while there are only 2363 and 4229 such variables for the DOS model. Clearly, this helps finding an optimal solution much more rapidly. One can observe that no branching is performed for instance $I_{A,14,202}$ even if there remains an optimality gap of 0.01%, which is less than or equal to the default tolerance of 0.01% accepted by CPLEX. One can observe that, for the first five instances, the cost of the computed solution is the same as that of the solution produced by the INT algorithm and differs by only one for the sixth instance. It shows that the constraints (Rules 3 to 8) that are relaxed to define the DOS model are not much binding because the large set of duties offers great flexibility to satisfy them.

Table 7: Results for the INT algorithm.

	#Variables	#Constraints	#Nodes	#Cuts	Time (s)	G^B (%)	G^A (%)	Cost
$I_{A,5,110}$	57645	20623	14285	226	2386	0.76	0	4966
$I_{A,6,124}$	51964	19844	19921	1931	5259	0.14	0	3461
$I_{A,10,146}$	98414	27342	16935	998	7005	0	0	6291
$I_{G,2,165}$	128537	47125	88147	480	41516	76.11	0	6222
$I_{G,2,180}$	152462	51409	80099	1232	40687	67.43	0	7853
$I_{A,14,202}$	189692	40691	17947	614	11969	0	0	18532
$I_{G,2,333}$	521813	99773	3	1082	> 43200	3.99	0.1	34682

Table 9 gives the results obtained in the duty assignment step of the TSM. These results are similar to those obtained for the first step: all instances are solved within short computational times except for the largest instance which required around 40 minutes. One can observe that, within the CPLEX tolerance, there is no integrality gap (even before adding cuts) for all instances. Nevertheless, some

cuts are needed to find an optimal solution, but no branch-and-bound nodes. With fixed days off, a relatively easy solution process was expected. Surprisingly, the costs of the solutions computed for the first and second steps are all equal for the first five instances and differ by one for the last two. This proves that the TSM has computed optimal solutions for all tested instances within the CPLEX optimality gap tolerance.

Table 8: Results of the days off scheduling step of the TSM.

Instance	#Variables	#Constraints	#Nodes	#Cuts	Time (s)	G^B (%)	G^A (%)	Cost
$I_{A,5,110}$	55727	2033	0	10	9	0.76	0	4966
$I_{A,6,124}$	50228	1912	0	51	23	0.14	0	3461
$I_{A,10,146}$	95950	2668	0	23	28	0	0	6291
$I_{G,2,165}$	125604	3070	0	133	111	76.11	0	6222
$I_{G,2,180}$	149263	3349	0	230	78	67.43	0	7853
$I_{A,14,202}$	186150	3725	0	3	49	0.01	0.01	18533
$I_{G,2,333}$	515891	6200	0	252	1001	3.98	0	13531

Table 9: Results of the duty assignment step of the TSM.

Instance	#Variables	#Constraints	#Nodes	#Cuts	Time (s)	G^B (%)	G^A (%)	Cost
$I_{A,5,110}$	40319	44461	0	39	15	0	0	4966
$I_{A,6,124}$	34791	40695	0	32	11	0	0	3461
$I_{A,10,146}$	67268	62229	0	66	42	0	0	6291
$I_{G,2,165}$	89809	102694	0	113	310	0	0	6222
$I_{G,2,180}$	106747	107416	0	79	168	0	0	7853
$I_{A,14,202}$	128715	113080	0	174	141	0	0	18532
$I_{G,2,333}$	352868	235891	0	101	2368	0	0	13531

In Table 10, we report the results obtained by the INTIS algorithm which considers the initial solution produced by the TSM. Note that the reported statistics exclude the computation of the initial solution. In particular, the computational time does not include the time for computing the initial solution. The total time will be discussed afterwards. From these results, we observe that providing high-quality initial solutions allow to solve to optimality the integrated model for all instances in less than three hours of computational time. Only the first instance requires very limited branching; the cuts completely close the gaps for four instances; no cuts, nor branching was needed to prove the optimality of the initial solution for two instances. Notice that the computational times are much smaller than those obtained without considering an initial solution (see Table 7).

Table 10: Results for the INTIS algorithm.

	#Variables	#Constraints	#Nodes	#Cuts	Time† (s)	G^B (%)	G^A (%)	Cost
$I_{A,5,110}$	57645	20623	6	392	486	0.76	0.76	4966
$I_{A,6,124}$	51964	19844	0	627	172	0.14	0	3461
$I_{A,10,146}$	98414	27342	0	0	14	0	0	6291
$I_{G,2,165}$	128537	47125	0	665	249	76.11	0	6222
$I_{G,2,180}$	152462	51409	0	642	3328	67.43	0	7853
$I_{A,14,202}$	189692	40691	0	0	53	0	0	18532
$I_{G,2,333}$	521813	99773	0	1223	8270	3.99	0	13532

†: Excluding the time required to compute the initial solution

Finally, let us compare the total computational times of the INT and INTIS algorithms, including the time required to compute the initial solution in the latter algorithm. These times are reported in Table 11 together with the gain in computational time (in percentage) achieved by the INTIS algorithm over the INT algorithm. The gain exceeds 90% for five of the seven instances and reaches a maximum of 99% for one instance. Obviously, these large speedups depend on the quality of the computed initial solutions.

Table 11: Total computational times for the INT and INTIS algorithms.

	Time (s)		
	INTIS	INT	Gain (%)
$I_{A,5,110}$	510	2386	79
$I_{A,6,124}$	206	5259	96
$I_{A,10,146}$	84	7005	99
$I_{G,2,165}$	670	41516	98
$I_{G,2,180}$	3574	40687	91
$I_{A,14,202}$	243	11969	98
$I_{G,2,333}$	11639	> 43200	> 73

5.3 Comparison with the current industrial solution algorithm

In this section, we compare the performance of the INTIS algorithm and that of the algorithm currently used by our industrial partner. Their algorithm, denoted IND hereafter, is a two-step matheuristic which also solves a days off scheduling problem first and a duty assignment problem second. In the first step, the problem is formulated as a mixed-integer linear program (see Gendron [8]) which is solved using a commercial MILP solver. Contrarily to the first step of the TSM, it involves no duty assignment variables. In the second step, the duty assignment problem is decomposed into a sequence of subproblems, each one being defined by a subset of positions and days. Each subproblem is formulated as a network-flow problem with side constraints and solved by a MILP solver. The subproblems are solved sequentially and, to ensure the continuity of the overall solution, constraints derived from the solutions of the subproblems previously solved may be imposed to a subproblem.

To compare the INTIS and IND algorithms, we report in Table 12 the costs of the solutions obtained by both algorithms on the full instances $I_{A,14,202}$ and $I_{G,2,333}$. This table also gives the gain (in percentage) obtained by the INTIS solution, the average workload per position, and the minimum and maximum workload over all positions for the solutions computed by both algorithms. Note that the solutions produced by the IND algorithm were computed in approximately one and three hours for the instances $I_{A,14,202}$ and $I_{G,2,333}$, respectively.

Table 12: Comparative results between the INTIS and IND algorithms.

	Cost			Workload (min.)		
	INTIS	IND	Gain (%)	Avg.	INTIS [Min, Max]	IND [Min, Max]
$I_{A,14,202}$	18532	19395	4.5	2113	[1995, 2400]	[1962, 2309]
$I_{G,2,333}$	13532	16173	16.3	2549	[2166, 2549]	[2137, 2621]

From these results, we observe significant gains realized by the proposed INTIS algorithm, namely, 4.5% (or 863 minutes) and 16.3% (or 2641 minutes) for the $I_{A,14,202}$ and $I_{G,2,333}$, respectively. Looking at the minimum and maximum workloads for the instance $I_{G,2,333}$, one can notice that the solution computed by the INTIS algorithm is better balanced than the IND solution, which is the goal of the CBDRP. For the other instance, the minimum and maximum workloads do not highlight this though the total number of minutes exceeding the average workload per position is much smaller for the INTIS solution.

6 Conclusion

In this paper, we addressed the CBDRP defined by our industrial partner and which does not involve shift-related rules but offers wide flexibility for duty assignment. We first modeled this problem as a mixed-integer linear program. Then, we proposed three algorithms to solve it. One is a two-step matheuristic, the other two consist of solving the integrated model using a MILP solver with or without an initial solution. Our computational experiments carried on two real-world instances and five additional instances derived from them showed that the two-step matheuristic yields, for these instances,

optimal solutions in relatively short computational times (up to 3.5 hours for instances involving up to 333 drivers). Furthermore, when providing these solutions as initial solutions to the MILP solver, very large speedups (up to 99%) were achieved to solve the CBDRP to proven optimality. Finally, we compared the computed optimal solutions with those produced by the matheuristic currently used by our industrial partner and observed significant gains in the solution cost, yielding a much better balanced workload between the positions.

Direct extensions of this work consist of designing efficient algorithms for solving the CBDRP with a variable number of positions per roster and the integrated duty scheduling and cyclic bus driver rostering problem in which the duties would be determined at the same time as the rosters.

References

[1] Bianco, L., Bielli, M. Mingozzi, A., Ricciardelli, & Spadoni, M. (1992). A heuristic procedure for the crew rostering problem. European Journal of Operational Research 58(2), 272–283.

[2] Borndörfer, R., Reuther, M., Schlechte, T., Schulz, C., Swarat, E., & Weider, S. (2015). Duty rostering in public transport – Facing preferences, fairness, and fatigue. ZIB-Report 15–44, Zuse Institute Berlin, Berlin.

[3] Caprara, A., Toth, P., Vigo, D., & Fischetti, M. (1998). Modeling and solving the crew rostering problem. Operations Research 46(6), 820–830.

[4] Carraresi, P., & Gallo, G. (1984). A multi-level bottleneck assignment approach to the bus drivers' rostering problem. European Journal of Operational Research 16(2), 163–173.

[5] Desaulniers, G., & Hickman, M. D. (2007). Public transit. In Handbooks in Operations Research and Management Science, Transportation Vol. 14, G. Laporte and C. Barnhart (eds.), Elsevier, Amsterdam, 69–127.

[6] Ernst, A.T., Jiang, H., Krishnamoorthy, M., & Sier, D. (2004). Staff scheduling and rostering: A review of applications, methods and models. European Journal of Operational Research 153(1), 3–27.

[7] Hartog, A., Huisman, D., Abbink, E. J., & Kroon, L. G. (2009). Decision support for crew rostering at NS. Public Transport 1(2), 121–133.

[8] Gendron, M. (2012). Détermination de la taille des effectifs et affectation des séquences de repos dans les horaires d'employés de compagnies de transport public. Master's thesis, Polytechnique Montréal, Montréal, Canada.

[9] Lezaun, M., Pérez, G., & Sáinz de la Maza, E. (2006). Crew rostering problem in a public transport company. Journal of the Operational Research Society 57(10), 1173–1179.

[10] Ma, J., Liu, T., & Zhang, W. (2014). A genetic algorithm approach to the balanced bus crew rostering problem. Journal of Traffic and Logistics Engineering 2(1), 13–20.

[11] 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. European Journal of Operational Research 245(2), 423–437.

[12] Moz, M., Respício, A., & Pato, M.V. (2009). Bi-objective evolutionary heuristics for bus driver rostering. Public Transport 1(3), 189–210.

[13] Nemhauser, G.L., & Wolsey, L.A. (1988). Integer and Combinatorial Optimization. John Wiley & Sons, New York.

[14] Nishi, T., Sugiyama, T., & Inuiguchi, M. (2014). Two-level decomposition algorithm for crew rostering problems with fair working condition. European Journal of Operational Research 237(2), 465–473.

[15] Nurmi, K., Kyngäs, J., & Post, G. (2011). Driver rostering for bus transit companies. Engineering Letters 19(2), 125–132.

[16] Sodhi, M. S., & Norris, S. (2004). A flexible, fast, and optimal modeling approach applied to crew rostering at London Underground. Annals of Operations Research 127(1–4), 259–281.

[17] Van Den Bergh, J., Beliën, J., De Bruecker, P., Demeulemeester, E., & De Boeck, L. (2013). Personnel scheduling: A literature review. European Journal of Operational Research 226(3), 367–385.

[18] Xie, L., & Suhl, L. (2014). Cyclic and non-cyclic crew rostering problems in public bus transit. OR Spectrum 37(1), 99–136.