Transportation Research Part E 156 (2021) 102530

Contents lists available at ScienceDirect

Transportation Research Part E

journal homepage: www.elsevier.com/locate/tre

Vehicle scheduling for rental-with-driver services

Simona Mancini [a,b,*], Margaretha Gansterer [a]

[a] University of Klagenfurt, Department of Operations, Energy, and Environmental Management, Universitätsstraße 65–67, 9020 Klagenfurt, Austria

[b] University of Eastern Piedmont, Department of Science, Innovation and Technology, Viale Teresa Michel 11, 15121 Alessandria, Italy

ARTICLE INFO

Keywords:
Vehicle scheduling
Combinatorial Benders cuts
Consistency
Workload limitations

ABSTRACT

In this paper, we introduce a new vehicle scheduling problem (VSP) with driver consistency faced by rental-with-driver companies. A weekly time-horizon is considered and a set of potential customers, each one associated with a list of required tasks, is assumed. The company can choose to accept or reject a customer, but if accepted, all required tasks must be performed by the same driver. A profit is associated with each customer. The goal is to maximize the company's total profit, by respecting a list of daily and the weekly drivers' workload limitations imposed by drivers' contracts. We propose a mathematical formulation of the problem and design an exact solution method based on the combinatorial Benders cuts approach.

A computational study based on several sets of instances reveals that the proposed solution method strongly outperforms the straightforward MIP approach. A deep analysis of the impact of different parameters is presented. Finally, we provide a measure of the cost of consistency, expressed as the loss of profit necessary to guarantee driver consistency. Results indicate that, for instance, if task durations are long, consistency can be achieved for almost no cost. However, if task durations are short, the loss of profit is below 6%. This provides an important managerial insight for companies offering luxury services, where the perceived quality of a service is key to its success.

1. Introduction

Rental-with-driver companies offer very customized services. Customers may ask for specific requirements, such as a driver speaking a specific language, specific vehicles (e.g., a limousine, suburban vehicles, luxury cars, or minivan), and special equipment such as DVD players. Although, in the past decades, this kind of luxury service was mainly devoted to transfer from airports to hotels and vice versa, in recent years the market for rental-with-driver services has expanded and started to serve several different categories of customers. One of these addresses business men and women, who come to the city for a fully-scheduled day and need a driver that takes them to different locations across the city. In this case, a stay-with policy is provided, according to which the driver is committed to the customer for the whole duration of the visit. This includes waiting time, e.g., meetings, to be ready to collect the customer as soon as desired. Stay-with tasks are generally characterized by a short travel distance but a long duration, which clearly impacts the price of the service.

Another category of customers that recently started using these services comprises elderly people who have to regularly visit, e.g., hospitalized relatives or residents of nursing homes. This category also includes people who must visit rehabilitation centers or hospitals frequently. Those requests are characterized by very frequently repeated specific tasks. Consistency in this case plays

* Corresponding author at: University of Klagenfurt, Department of Operations, Energy, and Environmental Management, Universitätsstraße 65–67, 9020 Klagenfurt, Austria.

E-mail addresses: simona.mancini@aau.at, simona.mancini@uniupo.it (S. Mancini), margaretha.gansterer@aau.at (M. Gansterer).

https://doi.org/10.1016/j.tre.2021.102530
Received 10 July 2021; Received in revised form 20 October 2021; Accepted 23 October 2021
Available online 20 November 2021

a crucial role, because such customers are typically old with limited mobility. Hence, they tend to establish a personal relationship with the driver, who is not only taking them to the destination, but also helping them to get into the car and to enter the facility. Such quality of services cannot be achieved using standard taxi services. Moreover, by signing a cumulative agreement with such a company, it is possible to obtain much lower prices with respect to the price demanded for a single task by a taxi driver. Differently from those related to the first category, these tasks are typically short. Medium range tasks are often requested by customers who have to reach an airport.

This setting raises some challenging optimization tasks in connection to vehicle and driver scheduling. Particularly, the complexity related to the consistency of these assignments to specific customers has not been tackled so far. Our study closes this research gap.

The contributions of our work can be summed up as follows:

1. We model a real-life vehicle scheduling problem (VSP) with driver–customer incompatibilities, arising in rental-with-driver companies, in which daily and weekly workload limitations occur.
2. While there are stacks of papers on consistency in vehicle *routing*, we are the first to consider consistency in vehicle-*scheduling* problems.
3. We provide a combinatorial Benders cut (CBC)-based exact method in which the master problem is formulated as a multi-dimensional multi-knapsack problem with incompatibilities.
4. The efficiency of the proposed algorithm is demonstrated through a wide set of experiments.
5. Managerial insights on the cost of consistency, i.e., the loss of profit necessary to guarantee driver consistency to customers requiring multiple tasks, are presented.

The paper is organized as follows. Section 2 presents related literature. A formal description of the problem is given in Section 3. The mathematical model is provided in Section 4, while an exact method based on CBC is described in Section 5. Computational results are analyzed in Section 6. Finally, we present our conclusions and discuss possible future research, in Section 7.

2. Literature review and methodological contribution

Despite the practical relevance, the problem of scheduling vehicles and drivers for a rental-with-driver company has generated limited interest in the operations research community. In Laurent and Hao (2007), the authors address the simultaneous vehicle and driver scheduling for a limousine company operating in the Paris area. The problem addressed in this paper is similar to ours but based on different assumptions. The authors consider a single-period problem, while in our model we address a multi-period horizon with workload constraints involving different periods simultaneously. Hence, the problem tackled in our study, cannot be split into several separated single-period problems. Furthermore, Laurent and Hao (2007) consider a dynamic environment in which some tasks are booked the day before but some others dynamically appear through the day. This assumption holds in specific contexts in which the company operates mostly on a direct service between airports and the city. In our study, we assume that rental-with driver services are generally booked at least one day in advance. Hence, a static environment seems to be more suitable. The underlying static problem does not include dynamically arriving requests; however, it is more complex to solve since it is working on a multi-period horizon, and we are considering driver workload constraints (daily and weekly), which are neglected in Laurent and Hao (2007). While the authors consider each task to be independent, we allow multiple tasks to be associated to the same customer. Furthermore, we ensure drivers' consistency for customers, i.e., we guarantee that all the tasks required by the same customer are operated by the same driver. This offers a high quality service for the customers. The rest of the literature review is organized as follows. In Section 2.1 we review the literature on the VSP, while Section 2.2 is focused on methodological aspects, particularly exact approaches based on CBC.

2.1. Vehicle scheduling problems

The VSP consists of assigning a set of timetabled trips, with fixed departure and arrival times and locations, to a set of vehicles. The objective is to minimize the overall costs, ensuring that each trip is executed and that each vehicle performs a feasible sequence of trips. Overall costs can be defined as a combination of fixed cost for vehicle usage and operational costs such as waiting times. This problem has been broadly addressed in the literature and many surveys on this topic are available. For an extensive review on VSPs, we refer the reader to Bunte and Kliewer (2009). Most of the papers in the literature study VSP arising in public transportation, in which, generally, the main goal is to minimize the number of vehicles required to fulfill all the tasks, while respecting workload constraints (Liu and Ceder, 2021). In this case, tasks are bus lines to be covered by a set of buses. Nevertheless, several applications of VSPs may also be found in the private sector such as airline scheduling in which a set of timetabled flights must be assigned to a set of aircrafts (Etschmaier and Mathaisel, 1985). In this context, the VSP is often integrated with the crew-scheduling problem (Papadakos, 2009). Since it is different from what occurs in airline scheduling – where aircraft and crew may have different schedules – in our application, the driver is associated with the vehicle; therefore, we do not include specific references about crew scheduling, but we refer interested readers to Deveci and Demirel (2018).

The basic version of the VSP deals with a set of trips starting and ending at the same depot (VSP-SD). This problem is relatively easy to solve, since it can be formulated as a problem for which polynomial time algorithms are known. The first study addressing the VSP-SD was presented by Saha (1972). The author defined a partial ordered set of trips, and computed the maximum cardinality set of pairwise incompatible trips, which equals the number of vehicles needed to cover all the trips. The problem is formulated as

a network flow problem. The main drawback of this approach is that it can only solve the problem of determining the minimum fleet size, but does not consider operational costs. In Orloff (1976), the VSP is modeled as an assignment problem on a complete bipartite graph. In this formulation, operational costs are considered but it is not possible to impose a fixed or a maximum number of vehicles to be used. In Gintner et al. (2005), a transportation model is presented, which has the advantage of providing, in case no feasible solution can be obtained with the given number of vehicles, a feasible schedule for vehicles and a list of unserved trips. A network flow model is provided in Bodin et al. (1983), while in Bertossi et al. (1987) the authors reformulate the VSP as a matching problem.

The multiple depot VSP addresses the case in which different depots for departure trips are available. An additional restriction is imposed that each vehicle must return to its starting depot at the end of its route. This problem has been proved to be $\mathcal{NP} - Hard$ in Bertossi et al. (1987).

A common extension of the VSP, which arises in practical applications, concerns the presence of multiple vehicle types (MVT). These are characterized by a potentially different fixed and/or operational cost. A limited number of vehicles is available for each type. The problem is shown to be $\mathcal{NP} - Hard$ even in the single depot case (Lenstra and Kan, 1981). A further extension, named VSP with vehicle type groups (VSP-VTG) considers the case in which a trip may be executed by only a subset of the vehicle types (Forbes et al., 1994). A network flow model for the MVT-VSP in which trips' starting times are not fixed but must be determined by the model within a feasible time window, is presented by Hassold and Ceder (2014). In Ceder (2011), the authors consider that for each trip the subset of compatible vehicle types is listed in a priority order based on their comfort level.

Since vehicles have to be associated with drivers, limitations due to workload regulations imposed by drivers' contracts may arise. To model this issue, an extension of the VSP, named VSP with route constraints (VSP-RC) has been introduced in Freling and Paixao (1995). In real world applications, limitations might apply not only for the daily but also for the weekly workload. Additional restrictions on lunch-break, minimum night rest, and maximum number of working days per week may also arise (Martello and Toth, 1986).

Most of the VSPs arise in a public transportation context. Typically, the goal is to minimize the number of vehicles needed to cover all the trips. In some studies, different types of vehicles are considered. In our problem, we address the case of a private company, which is not obliged to fulfill all the requests (i.e., to operate all the tasks), but it can select the subset of customers that maximizes its profit. This subset has to be selected such that a feasible schedule (e.g., in terms of drivers workload limitations, customers matching requests, and consistency) can be achieved with a given set of available drivers (and correspondent vehicles) characterized by different skills. To the best of our knowledge, this problem has not been treated so far.

Our problem shows some similarities with the Skill VRP (Cappanera et al., 2011; Schwarze and Voß, 2013) which aims at routing a set of technicians performing a set of tasks respecting task-technician compatibility. For instance, a technician must hold a minimum skill level to address a task. Nevertheless, our problem includes additional features which make it more complex to solve. First, we operate on a multi-period time horizon, in which both daily and weekly driving limitations occur, while the Skill VRP only considers a single period. Secondly, in our problem, tasks required by the same customer must be performed by the same driver, while in the Skill VRP each task is addressed as a single customer. Therefore, in the Skill VRP the problem of guaranteeing consistency does not hold. Thirdly, while in the Skill VRP all tasks have to be performed, and the goal is to minimize the total traveling cost, in our problem, each customer is associated with a profit and the goal is to maximize the total collected profit exploiting the current available resources. The company, however, is not obliged to accept all the customers.

While consistency is already researched in the literature on vehicle routing problems (VRP), it has not been tackled for VSPs so far. The concept of consistency was introduced by Groër et al. (2009), where two types of consistencies are defined; (i) time-consistency, according to which the customer must be served roughly at the same time at every visit and (ii) driver-consistency which imposes that a customer is visited always by the same driver. In the problem at hand, tasks' starting times are fixed in advance. Hence, only driver-consistency can be imposed. A generalized version of the consistent VRP, in which each customer can be visited by a limited number of drivers and the variation in the arrival times is penalized in the objective function, was introduced in Kovacs et al. (2015). For an extensive survey on consistency aspects in routing problems, we refer the reader to Kovacs et al. (2014). In some contexts, such as in routing of security patrols or highly valuable material transportation, inconsistency is required. In fact, in these cases, routes must not be predictable in order to prevent attacks from robbers. In Soriano et al. (2020), arrival time diversification on a multi-graph is discussed, while in Froehlich et al. (2020), inconsistency in the sequence of visits is addressed.

2.2. Combinatorial benders cuts

The Benders decomposition approach (BD) (Benders, 1962) is a well-known exact method to effectively solve mixed integer programming (MIP) models.

The core idea of BD is to split the original problem (OP) into a master problem (MP) and a sub problem (SP), which are sequentially solved following an iterative scheme.

In the first version of this method, proposed by Benders (1962), the SP is a linear programming (LP) problem, whose dual solution is exploited to derive cuts to be added to the MP. Later on, Geoffrion (1972) generalized BD and extended the method to cases in which the SP is not an LP.

Many decades later, a variant of the classical BD method, named logic-based Benders decomposition (LBBD), was proposed in Hooker and Ottoson (2003). In LBBD, variables directly contributing to the objective function are considered in the MP, while the remaining ones, which are responsible for feasibility, are relegated to the SP. This way, the SP turns out to be a pure feasibility problem. Whenever SP turns out to be infeasible, cuts are generated and added to the MP to cut off infeasible solutions from the

MP's search space. As soon as SP certifies that the solution provided by the MP is feasible, this solution is proved to be optimal also for the OP.

CBD is a specific case of LBBD and was addressed by Codato and Fischetti (2006). This approach is specifically designed for MIP models involving binary variables and a large number of logical implications. According to this approach, as soon as a specific combination of the MP variables provides a solution, which is certified by SP to be infeasible, a CBC is added to MP, to force at least one of the variables currently set to 1, to assume value 0. If the number of these variables is large, the obtained cut may turn out to be very weak. Stronger cuts can be obtained by detecting – optimally or heuristically – a subset of variables responsible for infeasibility. This is done through the identification of a minimum infeasible set (MIS). These cuts are stronger since they allow exclusion of several solutions from the MP search space simultaneously. Stronger cuts allow significantly speeding up of the convergence towards an optimal solution. However, it is important to remark that the convergence of the algorithm is proven even with only weak cuts (Codato and Fischetti, 2006). In fact, the optimal solution can always be reached in a finite number of steps. In the worst case, the number of cuts equals the number of feasible solutions of the MP minus 1. At the last iteration, the residual search space contains only one solution. If the SP certifies it to be feasible, this solution automatically becomes optimal for the original problem. Conversely, if the SP detect it to be infeasible, the OP can be certified to be infeasible, too.

In the last decade, CBC has been successfully applied to several real problems arising in different fields. The first application, described in Bai and Rubin (2009) concerns a facility location problem. Côté et al. (2014) proposed an exact approach based on CBC to effectively address the strip packing problem. Quayside operations optimization at container terminals are studied in Cao et al. (2010) and Chen et al. (2012), whereas Verstichel et al. (2015) address lock scheduling problems. An application in healthcare concerning beam intensity modulation in radiotherapy is studied by Taşkin and Cevik (2013). The method is applied to assembly line balancing problems by Akpinar et al. (2017). More recently, job allocation in computer clusters (Mancini et al., 2021) and multi-trip container drayage optimization (Bruglieri et al., 2021) were proposed. In Mancini et al. (2021), a new CBC framework has been introduced, where two important novelties from a methodological point of view are introduced. Firstly, the MP is not obtained by dropping some constraints from the OP, as in the classical CBC framework. Instead, the MP consists of solving a different (and easier to solve) combinatorial optimization problem. The second aspect of novelty concerns the MP's objective function, which is only an upper bound of the actual objective function. In this case, the SP's role is twofold: (i) it allows simultaneous checking of feasibility for the optimal solution of the MP and (ii) it computes the corresponding actual value of the objective function. If SP turns out to be infeasible, a standard CBC is added to the MP, while if the SP is feasible but the estimated objective function value was not correct, an additional penalty term is added to the MP objective function, which is activated only if the corresponding MP solution is selected. Also, Bruglieri et al. (2021) introduce some novel aspects from a methodological point of view. Although the authors exploit the classical CBC framework, i.e., obtaining the MP by dropping some constraints from the original formulation, they add constraints to the initial MP, to cut some solutions that are feasible for MP but that will certainly yield to an infeasible SP. This operation was proven to be able to strongly speed-up the solution process.

In our work, we also introduce novel aspects from a methodological point of view, which are used to create a new generalized CBC framework that can be exploited to address a large class of combinatorial optimization problems. The novelty of our approach is three-fold. The first novelty concerns the introduction of a preprocessing phase, composed of a list of several different incompatibility checks aiming at finding pairs of variables that cannot be simultaneously selected in a feasible solution. The MP is defined as a different combinatorial optimization problem, which is easier to solve with respect to the original problem, as proposed by Mancini et al. (2021). However, we add a set of CBCs to the initial MP to exclude the simultaneous selection of incompatible pairs of variables. This idea is based on the one exploited in Bruglieri et al. (2021) but extends and improves it, since the CBC constraints we propose, contain MIS's of a very small size (two elements). Hence, they are very strong cuts, which are able to cut off very large portions of the solution space. The second element of novelty is based on the fact that, in our case, infeasibility can be due to many different causes. Therefore, we design the SP such that it is able to detect not only if an infeasibility occurs but also to identify the cause. For each possible cause, we develop a different mechanism to find the corresponding MIS, allowing us to always provide very strong cuts to be added to the MP exploiting all the information we can retrieve from the execution of the SP. The last element of novelty concerns the fact that the SP does not need to solve another combinatorial optimization problem but consists of a dedicated algorithm performing a list of checks that can be executed in polynomial time. This allows to strongly speed-up the overall procedure. For all the above-mentioned reasons, we can state that we not only provide a solution method for the problem under investigation but also an effective new generalized CBC framework that can be exploited to address a broad class of combinatorial optimization problems.

3. Problem description

We address the customers-to-drivers assignment for rental-with-driver companies in a multi-period horizon. We assume that each driver is associated with a single vehicle. Customers book in advance and may request one or more timetabled tasks, which may start and end at different locations. Those tasks can require execution on the same day or on different days. To offer a premium service to their customers, companies guarantee that all the tasks of the same customers are carried out by the same driver. This is what we denote as *driver consistency*. Drivers adopt a stay-with policy, which means that they pick-up the customers at the departure location at a given time, accompany the customers in potential intermediate locations where they have to perform some activities and then bring them to the final destination at the agreed time. A single driver may serve an arbitrary number of customers on the same day, as long as their requests are not overlapping and the workload is compatible with her contract.

It is not mandatory to accept all the customers, but if a customer is accepted, all tasks required by this customer must be performed. A global price, for the entire set of tasks, is preliminarily agreed with the customer. The price might be influenced by different factors, such as the number of tasks required, the total travel distance, the total duration of the service, the fidelity of the customers (discounts are applied for frequent customers), and the specific requirements of the customers. When generating instances, we ensure that each customer can be served by the same driver, while drivers' workload constraints have to be respected. If this does not hold, the customer is virtually split into two or more fictitious customers, each one of which can be served by a single driver.

The problem can be formally described as follows. A multi-period horizon composed of D days is considered. A set of potential customers C and the related requests are known at the beginning of the time horizon. A set K of drivers, each one associated with a specific vehicle, is available. For each driver ($k \in K$), the company knows the home address (h_k) from which the driver starts and to which the driver returns after the daily shift. For each customer $c \in C$, the set of associated tasks, σ_c, and the total price p_c is known. Based on customers' specific requirements, a customer-driver compatibility matrix Φ is generated, such that $\phi_{ck} = 1$ if customer c and driver k are compatible, and 0 otherwise.

The total set of tasks is defined as I. For each task $i \in I$, the associated customer γ_i, the day on which it must be performed (δ_i), starting and ending locations (s_i and a_i), and starting and ending times (t_i^s, and t_i^a), are known. Consequently, the duration of task i (l_i) is computed as $t_i^a - t_i^s$. The travel time between each pair of locations is known as τ_{ij}. The daily working shift duration of driver k on day d, ST_{kd}, is identified as the time elapsed between the departure from and the arrival to the driver's home, while the daily working time, WT_{kd}, is defined as the sum of the duration of the tasks assigned to k on day d. The drivers contract imposes limits on the daily shift duration ST_{max}, daily working time WT_{max}, the weekly shift duration ST^w_{max}, and the weekly working time WT^w_{max}. The time elapsing between the end of a task and the departure to the starting point of the next task is called idle time. If the daily working shift duration is larger than a threshold α, at least one long break larger than a threshold β must be ensured. Furthermore, a minimum night break, ω, between the end of the shift on day d and the beginning of the shift on day $d+1$ must be ensured. Finally, each driver can work for at most ν days over the planning horizon. The goal of the problem is to accept and assign the subset of customers such that the company's profit is maximized, while consistency and drivers' workload constraints are respected. We assume that the company plans the driver–customer assignment on a weekly planning. It can be observed that these kind of services are typically not available on short notice.

4. Mathematical model

Let us define the following sets of decision variables and auxiliary variables.

X_{ijkd}	Binary variable taking value 1 if task j is executed immediately after task i by driver k on day d, 0 otherwise
Y_{ck}	Binary variable taking value 1 if customer c is assigned to driver k, 0 otherwise
Z_{kd}	Binary variable taking value 1 if driver k works on day d, 0 otherwise
Λ_{kd}	Binary variable taking value 1 if driver k is performing a long shift (i.e., longer than α) on day d, 0 otherwise
WT_{kd}	Non-negative variable representing working time for driver k on day d
ST_{kd}	Non-negative variable representing the shift duration of driver k on day d
T^{start}_{kd}	Non-negative variable representing the starting time for driver k shift on day d
T^{end}_{kd}	Non-negative variable representing the ending time of the shift of driver k day d
T_i	Non-negative variable representing the time at which the starting location for task i is reached
V_{kd}	Non-negative variable representing the longest break duration for driver k on day d

The mathematical model can be expressed as follows.

$$\max \sum_{k\in K}\sum_{c\in C} p_c Y_{ck} \tag{1}$$

$$\sum_{j\in J} X_{h_k jkd} = \sum_{j\in J} X_{j h_k kd} \quad \forall k \in K, d \in D \tag{2}$$

$$\sum_{j\in J} X_{h_k jkd} = Z_{kd} \quad \forall k \in K, d \in D \tag{3}$$

$$\sum_{j\in J \bigcup h_k} X_{ijkd} = \sum_{j\in J \bigcup h_k} X_{jikd} \quad \forall k \in K, d \in D \tag{4}$$

$$\sum_{j\in J} X_{ijk\delta_i} = Y_{\gamma_i k} \quad \forall k \in K \tag{5}$$

$$X_{ijkd} = 0 \quad \forall k \in K, i \in I, j \in I, d \in D : \; d \neq \delta_i \wedge d \neq \delta_j \tag{6}$$

$$\begin{aligned} T_j \geq t_i^a + \tau_{a_i s_j} - T_{max}(1 - X_{ij}) + \\ (T_{max} - \tau_{a_i s_j} + t_j^s - t_i^a) X_{ji} \quad \forall i \in I, j \in I \end{aligned} \tag{7}$$

$$T_j \leq t_j^s \quad \forall j \in I \tag{8}$$

$$T_{kd}^{start} \leq (t_i^s + \tau_{h_k s_i}) Y_{\gamma_i k} + T_{max}(1 - Y_{\gamma_i k}) \quad \forall i \in I : \delta_i = d, d \in D, k \in K \tag{9}$$

$$T_{kd}^{end} \geq (t_i^a + \tau_{a_i h_k}) Y_{\gamma_i k} \quad \forall i \in I : \delta_i = d, d \in D, k \in K \tag{10}$$

$$ST_{kd} = T_{kd}^{end} - T_{kd}^{start} \quad \forall i \in I : \delta_i = d, d \in D, k \in K \tag{11}$$

$$ST_{kd} \leq ST_{max} \quad \forall d \in D, k \in K \tag{12}$$

$$WT_{kd} = \sum_{i \in I : \delta_i = d} l_i Y_{\gamma_i k} \quad \forall d \in D, k \in K \tag{13}$$

$$WT_{kd} \leq WT_{max} \quad \forall d \in D, k \in K \tag{14}$$

$$\sum_{d \in D} ST_{kd} \leq ST_{max}^w \quad \forall k \in K \tag{15}$$

$$\sum_{d \in D} WT_{kd} \leq WT_{max}^w \quad \forall k \in K \tag{16}$$

$$V_{kd} \leq (t_j^s - t_i^a - \tau_{a_i s_j}) X_{ijkd} + T_{max}(1 - X_{ijkd}) \quad \forall i \in I, j \in I : \delta_j = \delta_i, d \in D, k \in K \tag{17}$$

$$\Lambda_{kd} \geq \frac{ST_{kd} - \alpha}{\alpha} \quad \forall d \in D, k \in K \tag{18}$$

$$V_{kd} \geq \beta \Lambda_{kd} \quad \forall d \in D, k \in K \tag{19}$$

$$\sum_{d \in D} Z_{kd} \leq \nu \quad \forall k \in K \tag{20}$$

$$T_{kd+1}^{start} - T_{kd}^{end} \geq \omega \quad \forall d \in \{1, \ldots, |D| - 1\}, k \in K \tag{21}$$

$$Y_{ck} \leq \phi_{ck} \quad \forall c \in C, k \in K \tag{22}$$

The objective function (1) maximizes the total profit. The combination of constraints (2) and (3) states that if a driver is on duty on a day, the shift must start from and end at their home location. Task sequencing is ensured by constraints (4). A task can be executed by a driver only if the customer who requested it has been assigned to that driver, as stated by constraints (5). Only tasks scheduled on the same day may be executed in sequence. This is formulated in constraints (6). The time on which the drivers reach the starting location of a task is tracked by constraints (7), and it must be earlier than the timetabled starting time of the task. The latter is ensured by constraints (8). It should be noted that constraints (7), which eliminate subtours, are strengthened as proposed by Yuan et al. (2020). Shift starting and ending time is computed, for each driver on each day, through constraints (9) and (10), respectively. Shift duration, calculated as in (11), cannot exceed the maximum allowed duration as given in constraints (12). Similarly, the total working time for a driver on a specific day must be lower than the maximum allowed working time, as specified by constraints (13) and (14). Restrictions on the maximum allowed shift duration and working time per week are imposed by constraints (15) and (16), respectively. Constraints (17) allow identification of the longest idle time for driver k on day d, while constraints (18) detect if driver k performs a long shift on day d. If this is the case, constraints (19) ensure that a minimum idle time must be guaranteed during the shift to schedule a break. A maximum number of working days per week is imposed by constraints (20). A minimum break must exist between two consecutive working shifts, as stated by constraints (21). Finally, a customer can be assigned to a driver, if and only if driver and customer are compatible as imposed by constraints (22).

5. Solution approach based on combinatorial Benders cuts

The proposed solution method consists of two phases. Firstly, a preprocessing procedure (see Section 5.1) is applied to identify pairs of customers that cannot be assigned to the same driver because the assignment would violate one or more driver workload constraints, or it would be infeasible due to an overlap of tasks. Different feasibility checks are applied in this phase. A pair is inserted in the incompatible pairs set, Ψ, if it violates at least one constraint. In a second phase, the CBC approach is applied. The MP (see Section 5.2) consists of solving a relaxation of the original problem, to which incompatibility constraints among pairs in Ψ are added to the formulation to forbid assignments that would yield to an infeasible SP. The SP becomes a pure feasibility check problem (see Section 5.3) in which we have to check constraints addressed neither in the preprocessing phase nor in the MP.

The role of the SP is twofold. Firstly, it checks the feasibility of the optimal solution of the MP. If feasible, then the solution is optimal for the original problem, too. If it is not feasible, the SP detects the causes of infeasibility and computes, by means of procedures designed ad-hoc for the cause of each type of infeasibility, a MIS (see Section 5.4). For each non-dominated detected MIS, a corresponding CBC is added to MP, which forbids the simultaneous selection of all the variables belonging to the MIS. A set A^1 is considered dominated by another set A^2 if $A^2 \subset A^1$. In fact, if, in this case, we impose that all the elements belonging to A^2 cannot be simultaneously picked, it holds automatically that they are simultaneously picked in combination with an additional set of items belonging to A^1 but not to A^2. These cuts are stronger than the classical CBC introduced by Codato and Fischetti (2006). The latter forbid simultaneously selecting all the variables active in the current MP's optimal solution. Our cuts, however, are able to exclude all the solutions in which the items belonging to the MIS are simultaneously selected. The cardinality of MIS being much smaller than the cardinality of the whole set of selected variables, the cut derived by MIS is able to cut-off a much larger number of solutions, and therefore is considered stronger. The MP and the SP are sequentially executed until the SP is found to be feasible.

This approach could be generalized to address all VSPs with consistency and also multi-task customers. It can also be applied in cases in which consistency is not required and/or in which only single-task customers are considered. In this case, the advantage in terms of reduced computational times would be smaller, since the number of customers equals the number of tasks, and consequently the assignment variables' number will be large. However, using the proposed approach would allow omission of the arc variables X_{ijkd}, which yields a strong global reduction in the number of variables. Therefore, the proposed CBC approach can be seen as a generalized framework for VSPs. A pseudocode of the proposed algorithm is reported in 1.

Algorithm 1 CBC pseudocode

```
Preprocessing phase
Add to MP constraints forbidding incompatible pairs of customers
Solve MP
Check MP optimal solution's feasibility through SP
if feasible then
  MP's optimal solution is optimal for the original problem too
else
  while not feasible do
    Detect cause of infeasibility
    for all causes of infeasibility do
      Add the corresponding cut to MP
    end for
    Eliminate duplicated and dominated cuts
    Run MP
  end while
  MP's optimal solution is optimal for the original problem too
end if
```

5.1. The preprocessing phase

The preprocessing phase aims to detect pairs of incompatible customers, i.e., customers that cannot be assigned to the same driver. A list of feasibility checks is defined. For each pair of customers, the checks in the list are executed sequentially. As soon as an incompatibility is detected, the two customers are marked as incompatible and further checks are skipped. To speed-up the checking process, the order in which checks are executed is based on the complexity of the checking algorithm, aiming at trying the fastest checks first, and moving on to more complex checking algorithms only if actually needed.

The set of compatibility checks exploited to detect conflicts between pairs of items are listed as follows:

1. If $w^w_{c^1} + w^w_{c^2} \geq WT^w_{max}$ then c^1 and c^2 are incompatible.
2. If for at least one day d, $w_{c^1 d} + w_{c^2 d} \geq WT_{max}$, then c^1 and c^2 are incompatible.
3. Let us define Ω_{c^1} and Ω_{c^2} as the set of days on which c^1 and c^2 request service, respectively. The two customers are incompatible if $|\Omega_{c^1} \bigcup \Omega_{c^2}| \rangle \nu$
4. If there exists a day d for which the difference between the ending time of the last task of a customer 1 (c^1) that has to be executed and the starting time of the first task to be executed on d of customer 2 (c^1), named $R_{c^1 c^2 d}$, is greater than ST_{max} then c^1 and c^2 are incompatible. Note that $R_{c^1 c^2 d}$ is a lower bound for the actual shift duration implied by serving both c^1 and c^2 since it does not take into account that the shift duration includes also transfer time from and to driver's home.
5. If the lower bound on the shift duration required over the whole week, to serve both c^1 and c^2, denoted as $R_{c^1 c^2}$ (where $R_{c^1 c^2} = \sum_{d \in D} R_{c^1 c^2 d}$) is greater than ST^w_{max}, then c^1 and c^2 are incompatible.
6. If there exists a day d for which the difference between the starting time of the first task of c^1 to be executed on day $d+1$ and the ending time of the last one to be executed on day d is greater than ω, then c^1 and c^2 are incompatible.

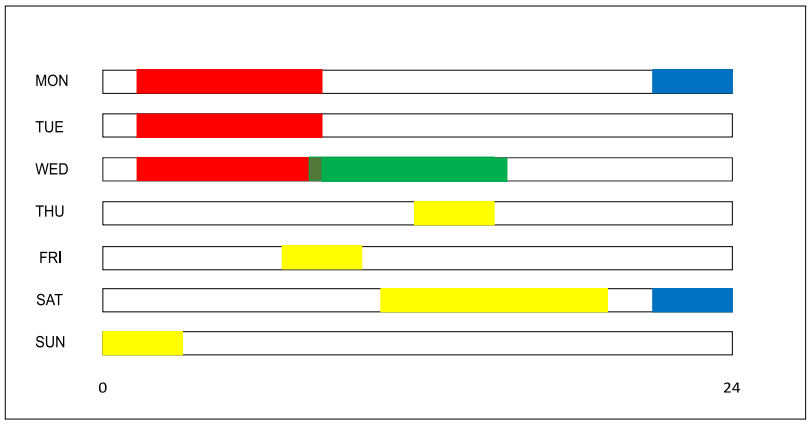

Fig. 1. An example for customer requests per week. Colors indicate that the requests belong to different customers. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

7. If c^1 and c^2 request tasks that are overlapping, they are incompatible.
8. If there exists a pair of tasks (i^1, i^2), one belonging to c^1 and the other one to c^2 such that the time needed to reach the starting location of i^1 from the ending location of i^2 ($\tau_{a_{i^1} s_{i^2}}$) is greater or equal to $t^s_{i^2} - t^a_{i^1}$, then c^1 and c^2 are incompatible.

For each pair of customers (c^1, c^2) these checks are carried out in a row. As soon as a check yields a negative outcome, i.e., an incompatibility is detected, the verification process is terminated and the pair is certified to be incompatible and added to Ψ. If a pair passes all the incompatibility checks, it is not inserted in Ψ.

A sample instance is depicted in Fig. 1. This instance contains four customers, requiring 10 tasks in total, spread across the whole week. The red customer and the green customer are incompatible because they have overlapping tasks on Wednesday. The red and the blue ones are incompatible because if a driver fulfilled both of them, the Monday shift would be too long. The red and the yellow are incompatible because a driver has to work too many days per week to fulfill both of them. All these incompatibilities would be detected in the preprocessing phase. Conversely, incompatibilities among more than two items can be identified only through the SP checking procedure, as for example, the incompatibility among the yellow, the blue and the green that cannot be assigned all to the same driver; otherwise, the maximum number of daily work in week 5 would not be respected. Assume that we have 2 drivers, which are all compatible with all customers, a feasible solution for this instance, could be the following. Driver 1 serves only the red customer, driver 2 serves both the blue and the yellow one, while the green customer is rejected.

5.2. The master problem

The relaxation used in the MP consists of modeling the problem as a multi-dimensional knapsack problem with conflicts (MDKPC), which is itself a new problem in the literature. It extends the knapsack problem with conflicts (KPC), addressed in Coniglio et al. (2021), by considering multiple dimensions for the knapsack. The KPC is a knapsack problem in which conflicts among items are considered. All pairs of items defined as incompatible cannot be simultaneously selected.

In our problem, each driver is modeled as a multi-dimensional knapsack, where a dimension is associated with each working day d, and an additional dimension represents the weekly workload. Each knapsack is associated with a capacity WT_{max} for all the first $|D|$ dimensions, while the capacity with respect to dimension $|D|+1$ is equal to WT^w_{max}. Each customer c is modeled as a single item, having weight w_{cd}, where $w_{cd} = \sum_{i \in I : \gamma_i = c \wedge \delta_i = d} l_i$ for the first $|D|$ dimensions, and weight $w^w_c = \sum_{i \in I : \gamma_i = c} l_i$ for dimension $|D|+1$. The conflicts between pairs of items are generated by means of the preprocessing phase.

The resulting master problem can be formulated as follows:

$$\max \sum_{k \in K} \sum_{c \in C} p_c Y_{ck} \tag{23}$$

$$\sum_{c \in C} w_{cd} Y_{ck} \leq WT_{max} \quad \forall d \in D, k \in K \tag{24}$$

$$\sum_{c \in C} w^w_c Y_{ck} \leq WT^w_{max} \quad \forall k \in K \tag{25}$$

$$Y_{ck} \leq \phi_{ck} \quad \forall c \in C, k \in K \tag{26}$$

$$Y_{c^1 k} + Y_{c^2 k} \leq 1 \quad \forall k \in K, c^1 \in C \wedge c^2 \in C : (c^1, c^2) \in \Psi \tag{27}$$

The objective function is reported in (23) and deals with profit maximization as in the original problem. Constraints (24) imply that, for each driver, the total time assigned on each day to serve the customers does not exceed the maximum allowed daily working time. Constraints (25) impose that the maximum weekly working time is respected for each driver. Each customer can be assigned only to compatible drivers, as ruled by constraints (26). Finally, constraints (27) forbid the simultaneous assignment of incompatible pairs of customers to the same driver.

5.3. Feasibility check and causes of infeasibility: the sub-problem

Note that the MP guarantees the feasibility of the schedule with respect to the maximum (daily and weekly) workload allowable, but not with respect to the maximum (daily and weekly) shift duration, the minimum break duration in long shifts, and the minimum night break. Therefore, it may happen that the schedule provided by the MP is infeasible for the OP.

The SP allows checking whether the solution is feasible, and if it is not, to identify the causes of the infeasibility. These causes can belong to the following groups.

IC1: Maximum daily shift duration exceeded. In the MP, only tasks' execution times are considered, while idle times between two consecutive tasks are neglected. Nevertheless, pairs of customers requiring service on the same day, whose requests are so far in time that the difference between the end of the second task and the beginning of the first one is larger than the maximum shift duration allowed, cannot be assigned to the same driver because they are marked as incompatible in the preprocessing phase (Check 4). However, the maximum daily shift duration may be exceeded by the MP optimal solution's schedule, since MP does not take into account the time necessary to reach the first task's starting point from the driver's home and the travel time from the last task's arrival point to the driver's home.

IC2: Maximum weekly cumulative shift duration exceeded. The MP implies that the maximum weekly workload is respected, but since it does not explicitly consider shifts' duration, it is possible that the optimal solution of the MP does not respect the maximum weekly cumulative shifts' duration allowed. This violation may occur even if the daily shift duration is respected for all the days. In fact, drivers' working contracts generally allow a long daily shift duration if combined with shorter shifts in the other days of the week.

IC3: Minimum night break not respected. In the MP, we are considering only the sum of workload associated with tasks but not the time of the day in which they are executed. Hence, it could happen that the difference in time between the end of the shift on day d and the start of the shift on day $d+1$ does not allow a sufficiently long night break. To partially avoid this issue, pairs of customers, whose tasks' schedule would lead to a too-short night break, are excluded in the preprocessing phase (Check 6). However, in Check 6, time needed to reach the driver's home from the ending location of the last task on that day and to reach the starting location of the first task of the next day is neglected. Therefore, a pair that is marked as feasible by Check 6 could be infeasible for some drivers.

IC4: Minimum break during a long shift not respected. Differently from the other causes of infeasibilities, this type can be detected in the preprocessing phase only in the case in which, on a specific day, a driver serves only two customers. Otherwise, since the duration of the (lunch) break does not depend only on the first and the last served in the day but also potentially on all the customers served on that day, this type of infeasibility cannot be detected analyzing only pairs of customers.

It should be noted that all the causes of infeasibilities may simultaneously apply, but none of them directly imply the occurrence of another one. Thus, they are neither strictly dependent nor correlated. The number of feasibility checks in the preprocessing is 8, while the number of infeasibility causes is only 4. This is due to the fact that some of the infeasibility checks in the preprocessing (namely 1,2,3, and 6) are exhaustive checks, while the remaining ones are approximated. Although they identify most of the incompatible pairs, they do not completely prevent the infeasibility of the MP optimal solution. However, even if they cannot guarantee feasibility, they strongly help to reduce the solution space of the MP, removing an exponential number of solutions, which are feasible for the MP but not for the OP, and make the MP a much tighter relaxation for the OP.

Differently from classical CBC approaches, in which a MIP model needs to be solved to check the feasibility of the MP solution, in our approach, feasibility can be checked through simple polynomial algorithms, speeding up the whole process. This is possible due to the fact that all the variables are already considered in the MP, and therefore, the solution obtained by solving the MP is already a complete solution of the original problem. Instead, in the classical CBC approach, the solution of MP is only a partial solution for OP, since not all the variables are considered in the MP. Typically, a set of them is only relegated to the SP.

5.4. Minimum infeasible sets detection

For each type of infeasibility, a different ad-hoc procedure to identify the MIS is exploited. All the procedures are given in the following.

IC1: Maximum daily shift duration. Among all tasks scheduled for driver k on day d for which the MP schedule exceeds the maximum daily shift duration, we define c_{kd}^{first} and c_{kd}^{last}, as the customers associated with the first and the last task, respectively. We add to the MP the following cut:

$$Y_{c_{kd}^{first}k} + Y_{c_{kd}^{last}k} \leq 1 \tag{28}$$

The set $\{c_{kd}^{first}, c_{kd}^{last}\}$ is a MIS, since the violation of the daily shift duration only depends on the first and the last customers served on this day. Adding constraints (28) to the MP, we are cutting-off at once all the solutions in which c_{kd}^{first} and c_{kd}^{last} are assigned to driver k, which is an exponential number.

IC2: Maximum weekly cumulative shift duration. For each driver k for which SP has detected an infeasibility on the weekly cumulative shift duration, the corresponding MIS is the set of customers containing the c_{kd}^{first} and c_{kd}^{last} for all the days in which the driver is operative, avoiding duplicates. We then impose that these customers cannot be simultaneously assigned to driver k, by adding to MP the following cut:

$$\sum_{c \in MIS} Y_{ck} \leq |MIS| - 1 \tag{29}$$

While constraint (28) is the strongest possible, since the associated MIS contains only two variables, constraint (29) is not proven to be the strongest possible, since there could be a subset of the current MIS responsible for the infeasibility. However, it is stronger than a classical CBC, which would involve all the customers assigned to k and not only those served first or last on a day. This particularly holds when the number of customers served by the same driver on a day is quite large, i.e., when tasks are generally short.

IC3: Minimum night break. For all the pairs of drivers and days (k, d) for which the MP does not ensure the minimum night break required, we impose that the last customer served by k on day d and the first served by k on day $d+1$ cannot be simultaneously assigned to k, adding to MP the following cut:

$$Y_{c_{kd+1}^{first}k} + Y_{c_{kd}^{last}k} \leq 1 \tag{30}$$

Again constraint (30) is the strongest possible cut since it is associated to a MIS of cardinality 2.

IC4: Minimum break during a long shift. For all the pairs of drivers and days (k, d) for which the MP does not ensure the minimum break required for a long shift, we impose that the customers served by k on day d cannot be simultaneously assigned to the driver. This is achieved by adding the following constraint:

$$\sum_{c \in MIS} Y_{ck} \leq |MIS| - 1, \tag{31}$$

where the MIS contains all the customers served by k on day d. This cut, similar to (29) is not proven to be the strongest possible, but ends up stronger than the classical CBC, since the latter would only involve customers who have to be served on day d and not all the customers assigned to k.

It should be noted that a cut is added to MP only if it is not duplicated. Furthermore, cuts strictly dominated by stronger ones are dropped. This happens, for instance, if an infeasibility implies that $c_1 + c_2 + c_3 \leq 2$ and another infeasibility implies that $c_1 + c_2 \leq 1$. In this case, the first cut is strongly dominated by the second one.

6. Computational results

In this section, we report the results obtained through a computational study comprising three parts: (i) comparison of the performances of the MIP model solved by a commercial solver (denoted as *model*) and the proposed CBC, (ii) analysis of the cost of consistency. To perform these analyses, seven sets of instances have been generated (S1–S7). Each set includes 10 instances sharing the same setting.

S1 includes instances with 100 tasks and 47 customers, 25% of which are single-task customers, while the remaining request multiple tasks. The length of the tasks is randomly drawn between 0.25 and 8 h. The number of drivers is tuned such that the total available working time accommodates 80% of the demand. The customer-driver compatibility matrix Φ is constructed such that around 75% of the assignments are feasible.

Instances in S2 contain the same tasks as those in S1, but Φ is decreased to 50%.

Sets S3 and S4 hold the same settings as S1 except that the number of drivers is able to cover only 50% and 20% of the total demand, respectively. While the first four sets share the same customers and tasks, in S5 all the tasks are short. They last between 0.25 and 1 h. The other settings, such as number of tasks and customers, percentage of single-task customers, and number of drivers and compatibility are the same as in S1. The difference between S5 and S6 lies in tasks' duration which, in S6, is randomly drawn between 0.25 and 3 h.

In S7, task duration follows the same distribution as in S6. The number of tasks is 100, as in all the other sets, but the number of customers is higher: 75, where 50% of which require a single task. S8 is based on S7 except that we consider a full driver–customer compatibility (every driver is compatible with every customer). S9 is also based on S7, but considers twice as many drivers. S10 contains larger instances with 200 tasks and 125 customers, 20% of which requesting multiple tasks. S11 also contains instances

Table 1
Data sets and general parameter settings.

Set	#Tasks	#Cust.	Single	Cap.	Comp.	Duration
S1	100	47	25%	80%	75%	[0.25–8]
S2	100	47	25%	80%	50%	[0.25–8]
S3	100	47	25%	50%	75%	[0.25–8]
S4	100	47	25%	20%	75%	[0.25–8]
S5	100	47	25%	80%	75%	[0.25–1]
S6	100	47	25%	80%	75%	[0.25–3]
S7	100	75	50%	80%	75%	[0.25–3]
S8	100	75	50%	80%	100%	[0.25–3]
S9	100	75	50%	160%	75%	[0.25–3]
S10	200	125	80%	80%	75%	[0.25–3]
S11	200	150	66%	80%	75%	[0.25–3]
S12	400	100	0%	80%	75%	[0.25–3]

Table 2
Parameters settings related to drivers' workload limitations.

WT_{max}	9
WT^{w}_{max}	32
ST_{max}	11
ST^{w}_{max}	40
ν	5
α	9
β	0.75
ω	8

Table 3
Comparison between model and CBC-based approach.

Set	Of	#ac	Model		CBC				
			T_F	T_{TOT}	#iter	#cuts	T_P	T_S	T_{TOT}
S1	1693.3	27.5	2.63	2.97	4.6	9.3	0.07	3.10	3.17
S2	1647.1	25.4	1.36	1.4	3.4	7.0	0.07	1.25	1.32
S3	1273.2	19.9	1.5	1.53	4.6	7.6	0.05	1.50	1.55
S4	676.6	10.1	0.38	0.42	1.3	0.7	0.07	0.12	0.19
S5	408.1	29.3	5.69	6.12	4.6	8.2	0.07	3.2	3.27
S6	797.7	27.5	5.26	6.26	3.6	9.6	0.05	2.75	2.80
S7	1437.1	67.4	196.01	196.06	12.3	41.5	0.09	35.42	35.51
S8	1464.3	67.1	422.06	422.12	10.4	38.2	0.10	33.60	33.70
S9	1436.8	67.8	47.28	47.36	6.5	18.4	0.10	1.28	1.38
S10	1509.4	90.4	44.65	44.76	15.7	46.3	0.16	2.24	2.40
S11	1483.3	80.5	145.70	145.83	9.3	36.4	0.15	12.30	12.45
S12	2082.9	11.3	10.23	11.12	3.5	9.7	0.81	2.18	2.99

with 200 tasks but with a larger number of customers (150) and a larger percentage of customers requiring multiple tasks (33%). Finally, S12 contains very large instances with 400 tasks and 100 customers, all requiring multiple tasks. Settings for all the sets of instances are listed in Table 1. Input parameters related to workload limitations are resumed in Table 2.

The model is solved by the commercial solver Xpress 7.9 with a time limit of 1 h. The CBC is implemented in the Xpress-Mosel language, and the MP is run with Xpress 7.9. A global time limit of 1 h is imposed for the whole CBC procedure.

All computational tests have been executed on a system equipped with an Intel-i7-5500U processor running at 2.4 GHz clock speed and with 16 GB of RAM. In Table 3, we report computational results obtained with the two approaches. Each row reports average values for the 10 instances of a set. We report the value of the optimal solution (OF) and the number of accepted customers (#ac). For the model, we also report the time at which the optimal solution was found, (T_F), and the time needed to solve the problem to optimality (T_{TOT}). For the CBC, we report the number of iterations required (# iter), the total number of cuts added (# cuts), the time needed for the preprocessing phase aiming at identifying incompatible pairs of customers (T_P), the time needed to solve the CBC (T_S), and the total time needed to solve the problem to optimality computed as $T_{TOT} = T_P + T_S$. Note that the CBC-based approach starts from an infeasible solution and reduces the solution space for the relaxed problem, until the optimal solution is feasible for the original problem, too. For this reason, as soon as a feasible solution is found by the SP, it is proven to be optimal. Therefore, the time at which the optimal solution is found (T_F) coincides with the total elapsed time T_{TOT}. All times are expressed in seconds. Detailed results are available online (Mancini and Gansterer, 2021).

Computational results show that both the compatibility and the number of drivers (expressed as a function of the ratio between capacity and demand) do not have a relevant impact on the difficulty of the instances. The latter is measured as the time needed to solve the instances to optimality and to find the optimal solution. Conversely, tasks' duration significantly influences computational times. In fact, on the first four sets in which tasks' duration have a very large variance, ranging from very small tasks to full-day

ones, both algorithms show very similar performances, requiring less than 4 s to solve all the instances to optimality. In sets S5 and S6, in which all tasks are short (less than 1 h in set S5 and less than 3 h in set S6), it is possible to notice a slight difference in the performances of the algorithms. While CBC takes about 3 s, the model requires 6 s. This difference, which is relatively small on sets S5 and S6, exponentially grows with the increase in the number of customers. In fact, when we pass from 47 customers to 75, as in set S7, computational times drastically increase for the model passing from 7 to 196 s (an increase of almost 30 times), while the increment is much smaller for CBC. More precisely, it increases by a factor of 10 from 3 to 35 s. The fact that instances with short tasks are more challenging is mainly due to the fact that the assignment problem is more difficult, since several tasks (and consequently several customers) can be assigned to the same driver, while, when dealing with long tasks, the number of tasks that can be performed by a driver on the same day is very limited (generally 2 or 3). Since the mathematical model can be seen as a VRP on an extended network in which tasks are considered as nodes and the sequence of tasks executed by a driver on a specific day can be seen as a route, shorter tasks allow for routes with a large number of visited nodes. It is known from the literature that VRPs in which it is possible to serve a large number of customers in the same route are much more challenging than those in which the number of customers per route is very limited. This explains why the model performs poorly when tasks are short. In the MP of the CBC the problem is, instead, modeled as a multi-dimensional multi-knapsack problem with side constraints. Conversely, knapsack problems do not suffer from the negative effect of having very small items with respect to the knapsack's capacity. On the contrary, problems in which a large number of items can be accommodated in the same knapsack, are generally less challenging than those in which only few items can be simultaneously assigned to the same knapsack. This explains why computational times required by CBC do not significantly increase when we move from long tasks to short ones.

While on the small instances with 47 customers, driver–customer compatibility does not affect computational times, we observe that when the number of customers grows to 75, this parameter has a strong impact on the performance of the model. CBC, on the other hand, is only slightly affected by this increase. In fact, passing from S8 to S7, having S8 the same characteristic as S7 except to a full compatibility matrix, average computational times for the model are doubled (from 196 to 422), while they are almost constant for CBC (from 35 to 33). Conversely, as predictable, a higher number of drivers (i.e., a higher capacity of the available resources) makes the problem easier to solve for both methods. Nevertheless, while the model is requiring 47 s on average, CBC only runs 1.38 s. Comparing results of S10 and S11, we can observe that the increment of the number of customers (from 125 to 150) and the number of those customers requiring multiple tasks (from 25 to 50), strongly impacts computational times of both methods. However, while computational times of the model increase from 44 to 145 s, CBC results to be considerably faster, requiring only 2.4 s (S10) and 12.45 s (S11).

The setting parameter which has the highest impact on the complexity seems to be the number of customers. However, as stated above, CBC is much less affected by this issue. The number of tasks is kept constant over all the instances. However, it is worth noticing that, while the number of tasks directly impacts the number of variables for the model, since the X_{ijkd} variables are defined for each pair of tasks, this does not directly impact CBC since, in this approach, the number of variables is only influenced by the number of customers and not by the number of tasks. The model becomes intractable when the number of tasks grows (assuming a fixed number of customers), while CBC marginally suffers from this issue. The only parameter which can significantly negatively impact CBC is the number of pairs of incompatible customers. In fact, although the feasibility check preprocessing is very fast (less than 0.1 s), a large number of incompatible pairs could be challenging to treat, since it would imply the addition of a large number of constraints that could slow down the MP-solving process.

To summarize, the model performs very well when tasks are long, while its performance significantly worsens when tasks are short and the number of customers increases. Conversely, CBC is only slightly affected by the presence of only short tasks. The parameter which mostly impacts its performance is the number of customers, but this impact is negligible compared to the model. As we can notice from Table 3, the number of iterations required to converge towards an optimal solution is very small, which means that (i) the MP provides a tight approximation of the original problem, and (ii) the cuts generated are very effective. The fact that the number of added cuts is larger than the number of iterations means that, on average, more than one cut per iteration is generated (i.e., more than one cause of infeasibility is detected by the SP). The newly proposed cuts allow speed-up of the method with respect to a classical CBC approach in which at each iteration only one cut is used.

To evaluate the cost of consistency, we relax the consistency constraints and compare the optimal profit to the one which guarantees consistency. For this purpose, we transform the instances associating each task with a different virtual customer. Since in the original instances, the profit is associated with the customers and not with single tasks, we share the profit among tasks associated with the same original customer, proportionally to the tasks' duration. An interesting insight is that, on instances with large tasks (S1–S4), consistency comes almost for free (the increment of profit achievable dropping consistency constraints is only 0.74%), while on instances with short tasks, we observe a cost of consistency of 5.59%. Such value seems to be acceptable for rental-with-driver companies, which typically want to offer high quality service and are willing to pay a reasonable cost to guarantee consistency.

7. Conclusions

In this paper, we presented a vehicle scheduling problem (VSP) arising in the context of rental-with-driver companies. This problem differs from most of the VSPs addressed in the literature for several reasons. Firstly, in the context we analyze, companies are not obliged to serve all the customers, while in VSPs arising in the context of public services or in freight delivery, all the tasks have to be performed. Another relevant aspect of novelty concerns the multi-period environment, in which drivers' contracts workload limitations are accounted for, limiting the daily and the weekly workload and inserting specific constraints regarding shift duration, minimum break for long shifts and minimum night breaks, as well as a limitation on the maximum number of working

days per week. The third element of novelty is the consideration of driver consistency for customers requiring more than one task. Although consistency has been broadly addressed in the vehicle-routing context, we are the first to integrate it in a vehicle scheduling context.

We propose a MIP model based on a virtual network representation in which nodes are tasks. Moreover, we propose an exact method based on the CBC approach. This method extends the classical CBC by defining the MP not as a version of the original problem, where some constraints have been dropped, but as a completely different combinatorial optimization problem, namely a multi-dimensional multi-knapsack problem, in which each customer is represented by a single item. The second element of novelty, from a methodological point of view, is that differently from the classical CBC approach, in which only variables directly contributing to the objective function are addressed in the MP while the remaining ones are relegated to the SP, in our approach all the variables are considered in the MP. Therefore, the solution of the MP is a complete solution for the original problem. In this case, the SP is not a combinatorial optimization problem, but simply a list of feasibility checks that can be executed in a very short time. This allows speed-up of the overall procedure.

Additionally, we presented a preprocessing phase that efficiently identifies pairs of incompatible customers, to strongly reduce the solutions space of the MP by cutting-off solutions already known to be infeasible.

We presented computational results on instances characterized by different parameter settings. Those results showed that while on easy instances both approaches are suitable, on more challenging instances, CBC strongly outperforms the model.

Finally, we presented an analysis of the additional profit achievable by relaxing the consistency constraints. Results indicate that on instances with longer tasks, consistency comes almost for free, while on instances with short tasks, the cost of consistency is 5.59%. This is assumed to bean acceptable increment of costs for companies offering luxury customer-oriented services.

Future developments could concern the generalization of the proposed CBC framework to address a broad class of VSPs. Such a framework could also be used to provide initial plans in a dynamic VSP, where parts of the tasks are known in advance while others are dynamically revealed during the day. Moreover, to effectively address very large instances, the method can be easily converted into a heuristic approach in which a state-of-the-art heuristic for multi-dimensional multi-knapsack can be adapted to efficiently provide a solution for the MP. From a methodological point of view, the idea of using the SP to detect the causes of infeasibilities rather than to check feasibility, and to provide specific minimum infeasible sets for each type of infeasibility, can be generalized to create a new CBC-based framework for a wide range of combinatorial optimization problems.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work is supported by FWF Austrian Science Fund: P 34502-N.

References

Akpinar, S., Elmi, A., Bektaş, T., 2017. Combinatorial Benders cuts for assembly line balancing problems with setups. European J. Oper. Res. 259 (2), 527–537.
Bai, L., Rubin, P.A., 2009. Combinatorial Benders cuts for the minimum tollbooth problem. Oper. Res. 57, 1510–1522.
Benders, J.F., 1962. Partitioning procedures for solving mixed-variables programming problems. Numer. Math. 4 (1), 238–252.
Bertossi, A., Carraresi, P., Gallo, G., 1987. On some matching problems arising in vehicle scheduling models. Networks 17, 271–281.
Bodin, L., B., G., Assad, A., Ball, M., 1983. Routing and scheduling of vehicles and crews: the state of the art. Comput. Oper. Res. 10 (2), 63–211.
Bruglieri, M., Mancini, S., Peruzzini, R., Pisacane, O., 2021. The multi-period multi-trip container drayage problem with release and due dates. Comput. Oper. Res. 125, 105102.
Bunte, S., Kliewer, N., 2009. An overview on vehicle scheduling models. Public Transp. 1, 299–317.
Cao, J., Lee, D.-H., Chen, J., Shi, Q., 2010. The integrated yard truck and yard crane scheduling problem: Benders' decomposition-based methods. Transp. Res. E 46, 344–353.
Cappanera, P., Gouveia, L., Scutellà, M.G., 2011. The skill vehicle routing problem. In: Pahl, J., Reiners, T., Voß, S. (Eds.), Network Optimization. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 354–364.
Ceder, A., 2011. Optimal multi-vehicle type transit timetabling and vehicle scheduling. Procedia Soc. Behav. Sci. 20, 19–30.
Chen, J., Lee, D.-H., Cao, J., 2012. A combinatorial Benders' cuts algorithm for the quayside operation problem at container terminals. Transp. Res. E 48, 266–275.
Codato, G., Fischetti, M., 2006. Combinatorial Benders' cuts for mixed-integer linear programming. Oper. Res. 54, 756–766.
Coniglio, S., Furini, F., San Segundo, P., 2021. A new combinatorial branch-and-bound algorithm for the knapsack problem with conflicts. European J. Oper. Res. 289, 435–455.
Côté, J.-F., Dell'Amico, M., Iori, M., 2014. Combinatorial Benders' cuts for the strip packing problem. Oper. Res. 62 (3), 643–661.
Deveci, M., Demirel, N., 2018. A survey of the literature on airline crew scheduling. Eng. Appl. Artif. Intell. 74, 54–69.
Etschmaier, M., Mathaisel, D., 1985. Airline scheduling: An overview. Transp. Sci. 19 (2), 105–188.
Forbes, M., Holt, J., Watts, A., 1994. An exact algorithm for multiple depot bus scheduling. European J. Oper. Res. 72, 115–124.
Freling, R., Paixao, J., 1995. Vehicle scheduling with time constraint. Lect. Notes Econ. Math. Syst. 430, 130–144.
Geoffrion, A., 1972. Generalized Benders decomposition. J. Optim. Theory Appl. 10, 237–260.
Gintner, V., Kliewer, N., Suhl, L., 2005. Solving large multiple-depot multiplevehicle-type bus scheduling problems in practice. OR Spectrum 27 (4), 507–523.
Groër, C., Golden, B.L., Wasil, E., 2009. The consistent vehicle routing problem. Manuf. Serv. Oper. Manage. 11 (4), 630–643.
Hassold, S., Ceder, A., 2014. Public transport vehicle scheduling featuring multiple vehicle types. Transp. Res. B 67, 129–143.
Hooker, J.N., Ottoson, G., 2003. Logic-based Benders decomposition. Math. Program. 96, 33–60.

Kovacs, A.A., Golden, B.L., Hartl, R.F., Parragh, S.N., 2014. Vehicle routing problems in which consistency considerations are important: A survey. Networks 64 (3), 192–213.
Kovacs, A.A., Golden, B.L., Hartl, R.F., Parragh, S.N., 2015. The generalized consistent vehicle routing problem. Transp. Sci. 49 (4), 796–815.
Laurent, B., Hao, J.-K., 2007. Simultaneous vehicle and driver scheduling: A case study in a limousine rental company. Comput. Ind. Eng. 53, 542–558.
Lenstra, J.K., Kan, A.H.G.R., 1981. Complexity of vehicle routing and scheduling problems. Networks 11 (2), 221–227.
Liu, T., Ceder, A., 2021. Research in public transport vehicle scheduling. In: Currie, G. (Ed.), Handbook of Public Transport Research. pp. 388–408.
Mancini, S., Ciavotta, M., Meloni, C., 2021. The multiple multidimensional knapsack with family-split penalties. European J. Oper. Res. 289 (3), 987–998.
Mancini, S., Gansterer, M., 2021. Detailed results for Vehicle scheduling for rental-with-driver services. Mendeley Data V1, http://dx.doi.org/10.17632/55x6z7r5bf.1.
Martello, S., Toth, P., 1986. A heuristic approach to the bus driver scheduling problem. European J. Oper. Res. 24 (1), 106–117.
Orloff, C.S., 1976. Route constrained fleet scheduling. Transp. Sci. 10 (2), 149–168.
Papadakos, N., 2009. Integrated airline scheduling. Comput. Oper. Res. 36, 176–195.
Saha, J.L., 1972. An algorithm for bus scheduling problems. Oper. Res. Q. 21 (4), 463–474.
Schwarze, S., Voß, S., 2013. Improved load balancing and resource utilization for the skill vehicle routing problem. Optim. Lett. 7 (8), 1805–1823.
Taşkin, Z., Cevik, M., 2013. Combinatorial Benders cuts for decomposing IMRT fluence maps using rectangular apertures. Comput. Oper. Res. 40 (9), 2178–2186.
Verstichel, J., Kinable, J., De Causmaecker, P., Vanden Berghe, G., 2015. A combinatorial Benders' decomposition for the lock scheduling problem. Comput. Oper. Res. 54, 117–128.
Yuan, Y., Cattaruzza, D., Ogier, M., Semet, F., 2020. A note on the lifted Miller-Tucker-Zemlin subtour elimination constraints for routing problems with time windows. Oper. Res. Lett. 48 (2), 167–169.