A branch-and-price algorithm for a routing problem with inbound and outbound requests

Maxime Agius, Nabil Absi, Dominique Feillet, Thierry Garaix

A branch-and-price algorithm for a routing problem with inbound and outbound requests

Maxime Agius[a,*], Nabil Absi[a], Dominique Feillet[a], Thierry Garaix[a]

[a]*Mines Saint-Etienne, Univ Clermont Auvergne, INP Clermont Auvergne, CNRS, UMR 6158 LIMOS, F - 42023 Saint-Etienne France*

Abstract

In this paper we present a new problem arising in the context of non-emergency transportation of patients. We consider a hospital (the depot) and a set of patients with a medical appointment. Patients require either to go from home to hospital (inbound request) or from hospital to home (outbound request). The problem can be addressed as a Pickup and Delivery Problem, but the fact that all transportation requests are connected to the depot also allows tackling it as a special Multi-Trip Vehicle Routing Problem. We adopt the second standpoint and call it the Multi-Trip Vehicle Routing Problem with Mixed Pickup and Delivery, and Release and Due dates. We propose a specialized branch-and-price algorithm and demonstrate computationally that our approach outperforms a classical branch-and-price algorithm based on the Pickup and Delivery Problem modeling. We also show how our algorithm can be adapted to the solution of the Vehicle Routing Problem with Simultaneous Pickup and Delivery and Time Windows, and obtain new optimal solutions on benchmark instances.

Keywords: Medical transportation, pickup and delivery, multi-trip vehicle routing, column generation.

1. Introduction

Since the seminal paper of Dantzig and Ramser (1959) that introduced the Vehicle Routing Problem (VRP), thousands of research papers dealing with different variants of the VRP have been published. A wide range of formulations and solution approaches have been proposed and allow to tackle many types of real-world applications for the transportation of goods or people. Readers are referred to the book by Toth and Vigo (2014) for a relatively recent overview of the field.

In this paper we are interested in the problem of transporting patients to hospitals for medical appointments, and back home after their appointments. Vehicle routing problems in the context of medical transportation have been widely studied, especially in the context of emergency services. Works dealing with non-emergency transportation, like ours, are scarcer. We consider a hospital and a set of patients with a medical appointment, that have either to be picked up at home and transported to the hospital or the other way around. Appointments define time constraints that we call release dates (time at which patients are available for transportation) and due dates (time at which they must be arrived at their destination). Vehicles are available at the hospital to achieve these tasks and for more efficiency, patients can be regrouped in the same vehicle, as long as time constraints and the vehicle capacity are satisfied.

*Corresponding author
Email address: maxime.agius@emse.fr (Maxime Agius)

Clearly, the studied problem is a special Pickup and Delivery Problem (PDP), with every patient being moved from a pickup point to a delivery point. As transportation requests concern persons, it is also a special Dial-A-Ride Problem (DARP). However, the fact that all transportation requests start or end at the hospital makes this PDP very specific. Actually, the problem can also be viewed as a special multi-trip VRP (MTVRP), with vehicles performing multiple trips from the depot (the hospital). We adopt this standpoint and demonstrate that it helps solving the problem more efficiently.

Seen as a multi-trip VRP, the problem, however, raises some important issues. First, trips combine two types of requests: inbound requests, for patients that will be picked-up at their home location and transported to the hospital, and outbound requests, for patients that are initially at the hospital and need to be transported back home. It implies a complex management of the vehicle capacity, similar to that encountered in PDPs. Second, operations at home locations are not only constrained in time by the release date (resp., due date) at this node, but also by the due date (resp., release date) at the hospital, which implies a complex time management. Given the existing literature, our problem can be identified as a Multi-Trip Vehicle Routing Problem with Mixed Pickup and Delivery, and Release and Due dates. In the following, we coin it MTMPD-RD.

The main contributions of this paper are threefold. First, we introduce a new variant of the VRP, the MTMPD-RD, which is motivated by the non-emergency patient transportation. Second, we present a branch-and-price algorithm specifically designed for this problem and we evaluate it through an extensive computational campaign; in particular we show the benefits of our approach against an off-the-shelf PDP branch-and-price algorithm and against a two-phase approach in which mono-trip routes are generated first and a trip-to-vehicle assignment procedure is applied second. Third, we show how our algorithm can be used to solve the Vehicle Routing Problem with Simultaneous Pickup and Delivery and Time Windows, and we close some benchmark instances for this problem.

The paper is organized as follows: Section 2 reviews the literature related to the MTMPD-RD. Section 3 describes the problem and provides a mathematical formulation. Section 4 describes the proposed solution method, a branch-and-price algorithm with a column generation to solve the linear relaxation. Section 5 describes the pricing problem used in the column generation algorithm. Section 6 details the used instances and the experiments and discusses obtained results. Finally, Section 7 concludes this paper, summarizing our findings and suggesting possible extensions to this work.

2. Literature review

In this section, we position the MTMPD-RD in the literature. As briefly introduced in Section 1, the MTMPD-RD is a variant of the PDP, and of the MTVRP. Also, it concerns the non-emergency transportation of patients and can be assimilated to a DARP. In the following, we respectively position our problem regarding the PDP, the MTVRP, the DARP and the non-emergency transportation literature.

Positioning regarding the PDP literature

PDP have met a considerable interest in the literature. For complete states of the art, interested readers are referred to Berbeglia et al. (2007) and Battarra et al. (2014).

Berbeglia et al. (2007) propose a classification of the different variants of the PDP. PDPs are classified according to the route structure to be considered and the type of demand. Each variant

can further be divided into two variants, a single vehicle and a multiple-vehicle variant. The three main classes classified according to the route structure are the many-to-many problems (M-M), the one-to-many-to-one problems (1-M-1), and the one-to-one problems (1-1). In M-M problems, a commodity may have multiple origins and destinations and a location may be the origin or destination of multiple commodities. This class is the least studied in the literature as it is the one with the least concrete applications. In 1-M-1 problems, some commodities are delivered from the depot to customers and other commodities are picked up at the customers' locations and returned back to the depot. The 1-M-1 has several applications. For example, in the beverage industry, full bottles are delivered to customers while empty ones are returned to the depot (Privé et al., 2006). In 1-1 problems, a commodity has a single origin and a single destination. This problem arises in the context of urban carrier operations and maritime shipping, among others, and also in the context of passenger transportation, where it is defined as a DARP (Cordeau and Laporte, 2003; Cordeau, 2006). A recent review (Molenbruch et al., 2017) provides a comprehensive classification of the DARP literature.

Clearly, the MTMPD-RD is classified within the 1-M-1 class, since commodities are transported either from the depot to customers or from customers to the depot. The 1-M-1 problems can be classified according to the demand type. If customers ask for a simultaneous service, *i.e.*, a combined service of pickup and delivery with at least one customer asking for both a pickup and a delivery, the problem is called the Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD). The VRPSPD is the most generic and studied variant of 1-M-1 problems. Interested readers are referred to Koç et al. (2020) for a complete literature review. If every customer requires either a pickup or a delivery, but not both, the problem is said single demand. The MTMPD-RD is a single-demand problem.

In the 1-M-1 with single demand, two classes can be derived depending on the way commodities are handled. When all deliveries must be served first followed by all pickups, the problem is said with backhaul. When pickups and deliveries can be mixed, the problem is said mixed (*e.g.*, Vehicle Routing Problem with Mixed Pickup and Delivery (VRPMPD), see for example Mosheiov (1998)). The VRPMPD is actually a special case of the VRPSPD in which no customer makes a combined demand. So a valid solution method for the VRPSPD is also valid for the VRPMPD.

Very few exact solution methods have been developed for the VRPSPD and the VRPMPD. Subramanian et al. (2013) propose a branch-and-cut-and-price algorithm to solve the VRPSPD. Instances with up to 100 customers are solved optimally. To the best of our knowledge, it is currently the best exact solution for the VRPSPD, as well as for the VRPMPD. The MTMPD-RD is a VRPMPD in which we add release and due dates and the possibility of performing multiple trips.

In the literature, the VRP with Simultaneous Pickup and Delivery and Time Windows (VRPSPDTW) is the closest problem to the MTMPD-RD. However, the VRPSPDTW does not allow multiple trips. To the best of our knowledge, Angelelli and Mansini (2002) were the first authors to propose a method to solve the VRPSPDTW and until now, they are the only ones who proposed an exact method. They developed a branch-and-price algorithm to solve the VRPSPDTW minimizing the total cost. They tested their method on modified Solomon's instances (from Solomon (1987), initially designed for the Vehicle Routing Problem with Time Windows (VRPTW)) with up to 20 customers.

Since then, several authors have proposed heuristics to solve the VRPSPDTW. In these heuristics, the VRPSPDTW is usually defined as a hierarchical bi-objective problem: a primary objective minimizes the number of vehicles used and a secondary objective minimizes the total distance. Wang and Chen (2012) introduce new instances from Solomon's instances with up to 100 customers and presents a genetic algorithm to solve the problem. Several other heuristics have been proposed and

evaluated on the same instances: a parallel simulated annealing algorithm in Wang et al. (2015), a Tabu Search algorithm in Shi et al. (2018), an Adaptive Large Neighborhood Search with Path Relinking in Hof and Schneider (2019), and a lexicographic-based two-stage algorithm in Shi et al. (2020).

Positioning regarding the MTVRP literature

The multi-trip Vehicle Routing Problem (MTVRP) was introduced in Fleischmann (1990), which argues on the necessity for vehicles to perform multiple trips when demands are high compared to vehicle capacities and travel times are short. As explained in the survey on multi-trip vehicle routing problems by Cattaruzza et al. (2016), the interest for these problems in the literature grew in the last years, essentially due to the development of new city logistic distribution systems and the usage of small eco-friendly vehicles.

In the literature, different naming have been used to refer to the use of multiple trips. In this paper we use the terms trip and route. A trip is a sequence of customer location visits starting and ending at the depot and without any intermediate stop at the depot. A route is a succession of one or several trips processed by a single vehicle.

Despite the gain of interest for the use of multiple trips, publications on exact methods remain limited. Koc and Karaoglan (2011) propose a branch-and-cut algorithm, while Mingozzi et al. (2013) describe an exact algorithm for the MTVRP based on two set-partitioning formulations and column-and-cut generation procedures for the linear relaxation. Hernandez et al. developed two branch-and-price algorithms: for the MTVRP with time windows (Hernandez et al., 2016) and for the MTVRP with time windows and limited trip duration (Hernandez et al., 2014). Hernandez et al. (2016) also compare route-based and trip-based formulations *i.e.*, formulations where columns define complete routes or single trips. Clearly, the latter limits the size of the search space in the pricing problem because the search is limited to trips instead of sequence of trips; but conversely, it complicates the master problem in which the selected trips have to be able to combine into a feasible set of routes. Computational experiments show a slight advantage for the trip-based formulation on some instances, at the price of a much more complex implementation. In this work we adopt the route-based model. More recently, Paradiso et al. (2020) report a seven-step solution framework that allows solving efficiently different variants of the MTVRP. The framework relies on the combination of many ideas: the concept of structure (roughly speaking, a structure is a trip whose departure time is free), the generation of a relevant set of structure thanks to column generation and thanks to different relaxations, a branch-and-cut algorithm to optimally combine structures, ng-routes, subset-row inequalities, among others.

Positioning regarding the DARP and the non-emergency transportation literature

The literature on non-emergency transport of patients is relatively poor, the emergency case having attracted more attention. Fogue et al. (2016) studie a problem that shares many similarities with ours. The study focuses on the inconveniences of patients with wheelchairs or on stretchers. It deals with several health facilities and the goal is to minimize both the average waiting time of patients and the ambulance usage. The authors proposed a two-phase heuristic: first, a genetic algorithm assigns requests to drivers, and second, a scheduling algorithm builds driver routes. The authors tested the algorithm on different scenarios obtained from a real ambulance company. The experiments show that the solution provided by the proposed algorithm outperforms human experts, reducing the average waiting time of patients and increasing the ambulance usage.

Lim et al. (2017) studie a non-emergency ambulance service met by Hong Kong public hospitals. The objective is threefold and hierarchical: first, maximizing the number of requests served; second,

minimizing the total cost and third, balancing the workload for the staff. The authors proposed a local search metaheuristic using a variable neighborhood descent procedure. They applied their algorithm to real-world instances. The results show that the algorithm outperforms the human experts, improving the number of requests served.

As previously stated, non-emergency transportation of patients can be defined as a DARP. Indeed, a DARP is a pickup and delivery problem with time windows for transportation of people with extra client convenience constraints. The standard definition of the DARP (Cordeau and Laporte, 2003) states that, the ride time of a user must not exceed a maximal duration L and routes must respect a maximum route duration. The MTMPD-RD distinguishes from this definition in three points. First, we propose a formulation with a release date at the pickup node and a due date at the delivery node instead of classic time windows and a constraint to limit ride time of a user. Second, in the MTMPD-RD route duration is not limited. Finally, the main specificity of the MTMPD-RD is that all requests are linked with the depot, either the pickup or the delivery node is the depot and multi-trip is allowed. Several variants of the DARP, however, show some similarities with our problem.

Parragh et al. (2012) include staff/accompanying persons which have restrictions on availability and working time. This enforces to come back regularly at the depot where are located the staff members. Other constraints can enforce to come back regularly at the depot, like lunch breaks or vehicles disinfection (Zhang et al., 2015; Liu et al., 2015). In these cases, multi-trip is allowed but requests are still characterized by a pickup and a delivery node.

Liu et al. (2015) point out that in real-world applications, several requests might have the depot as the pickup or delivery location. They adapted their model by allowing the pickup or drop-off operations on these requests with large time windows to be grouped together which reduces the size of the graph and, the service time at the new node is equal to the sum of the concerned clients' service time.

Qu and Bard (2013) present a problem motivated by the route planning for Program of All-Inclusive Care for the Elderly (PACE) organisation. It consists in transporting elderly people between their home and activity centers for socializing, or to go to a medical appointment. The main specificity of the problem is that patients have different mobility type, so they might require a simple seat, or more space for a walker or a wheelchair. Hence PACE owns different types of vehicles whose interior can be modified. The problem is defined as a heterogeneous Pickup and Delivery Problem with configurable vehicle capacity. The paper presents a Mixed Integer Linear Program (MILP) formulation of the problem and an Adaptive Large Neighborhood Search algorithm. The same authors extended their work in Qu and Bard (2015) providing a branch-and-price-and-cut algorithm. The algorithm is tested on real instances and randomly generated instances and can solve instances with up to 50 customers to optimality. Other DARP variant problems are solved with exact methods (Cordeau, 2006; Luo et al., 2019).

3. Problem description and mathematical formulation

3.1. Problem description

The goal of the problem is to optimize the way customers are transported to or from a central depot on a time horizon $[0, T_{max}]$. This depot is denoted v_0 and customer locations v_1 to v_n. When a customer needs to be transported from their location to the depot, we call it an inbound request. Otherwise, when the customer is transported from the depot to their location, it is called an outbound request. We denote by $\mathcal{N}^+$ the set of inbound requests and by $\mathcal{N}^-$ the set of outbound requests. $\mathcal{N} = \mathcal{N}^+ \cup \mathcal{N}^-$ is the union of these two sets. We also equivalently denote $\mathcal{N} = \{1, \ldots, n\}$,

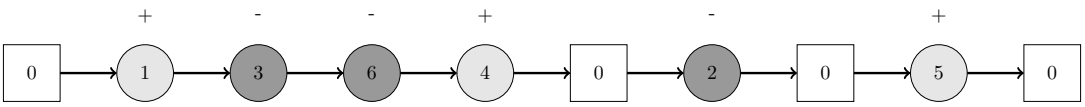

Figure 1: Route timing example

where the customer location of request i is v_i. In what follows, terms request, patient and customer will be used indifferently.

With each request $i \in \mathcal{N}$ is associated a demand $q_i \geq 0$, a release date r_i, a due date d_i and a service time s_i. The demand indicates the number of people involved in the request. The release date is the time at which these people are available at the pickup node. The due date is the latest allowed arrival time at the delivery node. The service time is the time needed at the customer location for pickup (inbound request) or drop off (outbound request). A time might also be spent at the depot, for pickups and drop-offs as explained below.

The transportation network can be represented by a complete graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, where $\mathcal{V} = \{v_0, v_1, \ldots, v_n\}$ and $\mathcal{A} = \{(v_i, v_j) | v_i, v_j \in \mathcal{V}, i \neq j\}$. A travel time t_{ij} and a travel cost c_{ij} are defined on each arc $(v_i, v_j) \in \mathcal{A}$. A homogeneous fleet $\mathcal{M} = \{m_1, \ldots, m_K\}$ of K vehicles of capacity Q is available to satisfy transportation requests. Vehicle routes consist of one or several successive trips, starting and coming back to the depot. Each trip is allowed to mix inbound and outbound requests. When leaving the depot (starting a new trip), a vehicle can either be empty (the trip will not fulfill any outbound request) or not. In the second case, a constant pickup time T_P is counted, before the vehicle can leave and after all release dates associated with the outbound requests served in the trip are passed. When returning to the depot (ending a trip), a vehicle can again either be empty (the trip did not fulfill any inbound request) or not. In the second case, a constant time T_D is counted for drop-off.

Figure 1 illustrates on a simple example the time constraints that have to be satisfied. Signs + and − respectively indicate inbound and outbound requests. The route is made of three trips. In Trip 1, the vehicle has to serve two outbound requests for patients 3 and 6. The vehicle thus leaves the depot not before time $\max(r_3, r_6) + T_P$, when the two patients are ready and picked-up. It first goes to the location of inbound patient 1. If it arrives earlier than r_1, it waits until the patient is ready (time r_1). Then, a time s_1 is spent for pickup. The vehicle then successively goes to locations v_3 and v_6 for drop-offs. These locations respectively need to be reached before due dates d_3 and d_6. Once these locations reached, drop-offs times s_3 and s_6 are respectively spent. At worst, the trip continues from v_6 at time $d_6 + s_6$. Then, patient 4 is picked-up (not before r_4, with service time s_4) and the vehicle returns to the depot with patients 1 and 4 inside. Due dates for these patients impose that the depot is reached before time $\min(d_1, d_4)$. When the depot is reached, the two patients leave the vehicle, for a drop-off time T_D. The vehicle is now empty and ready for a second trip. Trip 2 serves only one outbound request (patient 2), so the vehicle might have to wait for time $r_2 + T_P$ before starting. After having dropped-off patient 2 (before d_2, with service time s_2), the vehicle returns to the depot empty. The arrival time is only constrained by T_{max} and no drop-off time is incurred. Furthermore, as Trip 3 only contains an inbound request (patient 5), the vehicle can restart immediately. Patient 5 is picked-up (not before r_5, with service time s_5), the vehicle returns to the depot before d_5, a drop-off time T_D is added and the vehicle route is finished, at worst at time T_{max}.

A solution is feasible if it contains at most K routes, serves all requests and is such that all routes are valid, where a route r is valid if:

- r starts and ends at the depot (with one or several trips),
- the vehicle capacity is never exceeded in the route,
- the timing of the route is consistent with release dates, due dates, service times and the time horizon, as illustrated with Figure 1.

The objective is to minimize the total travelling cost. Table 1 reports the notation introduced in the definition of the MTMPD-RD.

In the following, we assume that instances are not trivially unfeasible. In particular, $r_k + s_k + t_{k0} \leq d_k$ holds for all inbound customers and $r_k + T_P + t_{0k} \leq d_k$ holds for all outbound customers. Furthermore, for inbound customers, we assume that $d_k \leq T_{max} - T_D$ to leave time for drop-off before the end of the time horizon. Finally, we assume that the triangle inequality holds for the travel time and the travel cost matrices.

Table 1: Notation

Sets	
$\mathcal{N} = \{1, \ldots, n\}$	set of requests
$\mathcal{V} = \{v_0, v_1, \ldots, v_n\}$	vertices of the graph
$\mathcal{A} = \{(v_i, v_j) \vert v_i, v_j \in \mathcal{V}, i \neq j\}$	set of arcs
$\mathcal{G} = (\mathcal{V}, \mathcal{A})$	transportation network
$\mathcal{M} = \{m_1, \ldots, m_K\}$	fleet of vehicles
$\mathcal{N}^+$	set of inbound requests
$\mathcal{N}^-$	set of outbound requests
Parameters	
n	number of requests
K	number of vehicles
Q	vehicle capacity
q_i	demand of customer i
r_i	release date of customer i
d_i	due date of customer i
t_{ij}	travel time between nodes v_i and v_j
c_{ij}	travel cost between nodes v_i and v_j
s_i	service time at location v_i of customer i
T_P	pickup service time at the depot
T_D	delivery service time at the depot
T_{max}	ending time of the horizon

3.2. Mathematical formulation

We now propose a mathematical formulation for the MTMPD-RD. Usually in the literature, 1-M-1 pickup and delivery problems are formulated either using a commodity flow model or a vehicle flow model (Koç et al., 2020). Instead of using these formulations, we adapt the vehicle flow formulation presented in Cattaruzza et al. (2016) for the MTVRP.

In order to represent the trips, we introduce set $\mathcal{H} = \{0, \ldots, n-1\}$ to number trips within a vehicle route. In the worst case, a vehicle can use up to n trips. The decision variables are reported in Table 2. Note that for time variables τ_i^{kt} and for load variables u_i^{kt}, we introduce a duplicate of the depot, v_{n+1}, to differentiate between the start and the end of the trip.

Table 2: Decision variables

Variables	
$x_{ij}^{kt} \in \{0,1\}$	1 if trip $t \in \mathcal{H}$ of vehicle $m_k \in \mathcal{M}$ uses arc $(v_i, v_j) \in \mathcal{A}$, 0 otherwise
$y_i^{kt} \in \{0,1\}$	1 if trip $t \in \mathcal{H}$ of vehicle $m_k \in \mathcal{M}$ serves request $i \in \mathcal{N}$, 0 otherwise
$i^{kt} \in \{0,1\}$	1 if at least one inbound request is served in trip $t \in \mathcal{H}$ of vehicle $m_k \in \mathcal{M}$, 0 otherwise
$o^{kt} \in \{0,1\}$	1 if at least one outbound request is served in trip $t \in \mathcal{H}$ of vehicle $m_k \in \mathcal{M}$, 0 otherwise
$\tau_i^{kt} \geq 0$	time at which trip $t \in \mathcal{H}$ of vehicle $m_k \in \mathcal{M}$ starts service at node $v_i \in \mathcal{V} \cup \{v_{n+1}\}$
$u_i^{kt} \geq 0$	load of vehicle $m_k \in \mathcal{M}$ on trip $t \in \mathcal{H}$ leaving node $v_i \in \mathcal{V} \cup \{v_{n+1}\}$

Using parameters and variables described in Tables 1 and 2, we propose the following mathematical formulation to model the MTMPD-RD.

$$\min \sum_{(v_i,v_j)\in\mathcal{A}} c_{ij} \sum_{m_k\in\mathcal{M}} \sum_{t\in\mathcal{H}} x_{ij}^{kt} \tag{1}$$

subject to

$$\sum_{m_k \in \mathcal{M}} \sum_{t \in \mathcal{H}} y_i^{kt} = 1 \qquad \forall i \in \mathcal{N} \quad (2)$$

$$\sum_{v_j \in \mathcal{V} \setminus \{v_i\}} x_{ji}^{kt} = y_i^{kt} \qquad \forall i \in \mathcal{N}, \forall m_k \in \mathcal{M}, \forall t \in \mathcal{H} \quad (3)$$

$$\sum_{v_j \in \mathcal{V} \setminus \{v_i\}} x_{ij}^{kt} = y_i^{kt} \qquad \forall i \in \mathcal{N}, \forall m_k \in \mathcal{M}, \forall t \in \mathcal{H} \quad (4)$$

$$\sum_{v_j \in \mathcal{V} \setminus \{v_0\}} x_{0j}^{kt} \leq 1 \qquad \forall m_k \in \mathcal{M},\ \forall t \in \mathcal{H} \quad (5)$$

$$\sum_{v_i \in \mathcal{V} \setminus \{v_0\}} x_{i0}^{kt} \leq 1 \qquad \forall m_k \in \mathcal{M},\ \forall t \in \mathcal{H} \quad (6)$$

$$i^{kt} \leq \sum_{i \in \mathcal{N}^+} y_i^{kt} \leq |\mathcal{N}^+| i^{kt} \qquad \forall m_k \in \mathcal{M},\ \forall t \in \mathcal{H} \quad (7)$$

$$o^{kt} \leq \sum_{i \in \mathcal{N}^-} y_i^{kt} \leq |\mathcal{N}^-| o^{kt} \qquad \forall m_k \in \mathcal{M},\ \forall t \in \mathcal{H} \quad (8)$$

$$u_0^{kt} = \sum_{i \in \mathcal{N}^-} q_i \times y_i^{kt} \qquad \forall m_k \in \mathcal{M},\ \forall t \in \mathcal{H} \quad (9)$$

$$u_j^{kt} \geq u_i^{kt} + q_j - Q(1 - x_{ij}^{kt}) \qquad \forall (v_i, v_j) \in \mathcal{A},\ j \in \mathcal{N}^+,\ \forall m_k \in \mathcal{M},\ \forall t \in \mathcal{H} \quad (10)$$

$$u_j^{kt} \geq u_i^{kt} - q_j - Q(1 - x_{ij}^{kt}) \qquad \forall (v_i, v_j) \in \mathcal{A},\ j \in \mathcal{N}^-,\ \forall m_k \in \mathcal{M},\ \forall t \in \mathcal{H} \quad (11)$$

$$u_{n+1}^{kt} = \sum_{i \in \mathcal{N}^+} q_i \times y_i^{kt} \qquad \forall m_k \in \mathcal{M},\ \forall t \in \mathcal{H} \quad (12)$$

$$u_i^{kt} \leq Q \qquad \forall v_i \in \mathcal{V} \cup \{v_{n+1}\},\ \forall m_k \in \mathcal{M},\ \forall t \in \mathcal{H} \quad (13)$$

$$\tau_j^{kt} \geq \tau_0^{kt} + T_P \times o^{kt} + t_{0j} - T_{max}(1 - x_{0j}^{kt}) \qquad \forall (v_0, v_j) \in \mathcal{A},\ \forall m_k \in \mathcal{M},\ \forall t \in \mathcal{H} \quad (14)$$

$$\tau_j^{kt} \geq \tau_i^{kt} + s_i + t_{ij} - T_{max}(1 - x_{ij}^{kt}) \qquad \forall (v_i, v_j) \in \mathcal{A},\ i \neq 0, j \neq 0,\ \forall m_k \in \mathcal{M},\ \forall t \in \mathcal{H} \quad (15)$$

$$\tau_{n+1}^{kt} \geq \tau_i^{kt} + s_i + t_{i0} - T_{max}(1 - x_{i0}^{kt}) \qquad \forall (v_i, v_0) \in \mathcal{A},\ \forall m_k \in \mathcal{M},\ \forall t \in \mathcal{H} \quad (16)$$

$$\tau_{n+1}^{kt} \geq \tau_0^{kt} \qquad \forall m_k \in \mathcal{M},\ \forall t \in \mathcal{H} \quad (17)$$

$$\tau_0^{k,t+1} \geq \tau_{n+1}^{kt} + T_D \times i^{kt} \qquad \forall m_k \in \mathcal{M},\ \forall t \in \mathcal{H} \setminus \{n-1\} \quad (18)$$

$$T_{max} \geq \tau_{n+1}^{k,n-1} + T_D \times i^{k,n-1} \qquad \forall m_k \in \mathcal{M} \quad (19)$$

$$\tau_i^{kt} \geq r_i \times y_i^{kt} \qquad \forall i \in \mathcal{N}^+,\ \forall m_k \in \mathcal{M},\ \forall t \in \mathcal{H} \quad (20)$$

$$\tau_{n+1}^{kt} \leq d_i + T_{max}(1 - y_i^{kt}) \qquad \forall i \in \mathcal{N}^+,\ \forall m_k \in \mathcal{M},\ \forall t \in \mathcal{H} \quad (21)$$

$$\tau_0^{kt} \geq r_i \times y_i^{kt} \qquad \forall i \in \mathcal{N}^-,\ \forall m_k \in \mathcal{M},\ \forall t \in \mathcal{H} \quad (22)$$

$$\tau_i^{kt} \leq d_i + T_{max}(1 - y_i^{kt}) \qquad \forall i \in \mathcal{N}^-,\ \forall m_k \in \mathcal{M},\ \forall t \in \mathcal{H} \quad (23)$$

$$x_{ij}^{kt} \in \{0, 1\} \qquad \forall (v_i, v_j) \in \mathcal{A}, \forall m_k \in \mathcal{M},\ \forall t \in \mathcal{H} \quad (24)$$

$$y_i^{kt} \in \{0, 1\} \qquad \forall i \in \mathcal{N},\ \forall m_k \in \mathcal{M},\ \forall t \in \mathcal{H} \quad (25)$$

$$i^{kt} \in \{0, 1\} \qquad \forall m_k \in \mathcal{M},\ \forall t \in \mathcal{H} \quad (26)$$

$$o^{kt} \in \{0, 1\} \qquad \forall m_k \in \mathcal{M},\ \forall t \in \mathcal{H} \quad (27)$$

$$0 \leq \tau_i^{kt} \leq T_{max} \qquad \forall v_i \in \mathcal{V} \cup \{v_{n+1}\},\ \forall m_k \in \mathcal{M},\ \forall t \in \mathcal{H} \quad (28)$$

$$u_i^{kt} \geq 0 \qquad \forall v_i \in \mathcal{V} \cup \{v_{n+1}\},\ \forall m_k \in \mathcal{M},\ \forall t \in \mathcal{H} \quad (29)$$

The objective function (1) minimizes the total cost. Constraints (2) ensure that each request

is assigned to exactly one vehicle and one trip. Constraints (3) and (4) are the flow conservation constraints. Constraints (5) and (6) ensure that trips start and end at the depot. Constraints (7) (resp., constraints (8)) ensure i^{kt} (resp., o^{kt}) equals to 1 when at least one inbound (resp., outbound) request is served in the trip, and equals to 0 otherwise. Constraints (9) to (13) are flow capacity constraints, where constraints (9) ensure that the load of a vehicle starting a trip is equal to the sum of loads of the outbound requests served in the trip, constraints (10) and (11) ensure vehicle load flow conservation when visiting an inbound (resp., outbound) customer, constraints (12) ensure that the load of a vehicle ending a trip is equal to the sum of loads of the inbound requests served in the trip, and constraints (13) check that vehicle capacity is never exceeded. Time precedence relations are ensured with constraints (14) to (19). Constraints (14) ensure that the pickup service time T_P is added when leaving the depot if the trip contains at least one outbound request. Constraints (15) model travel and service times between two customers while constraints (16) model the return to the depot. Constraints (17) ensure that the ending time of a trip is later than its starting time. Constraints (18) are used to add the delivery service time T_D when returning to the depot if the trip contained at least one inbound request. Constraints (19) ensure that the service of a route ends before the end of the time horizon. Release and due date constraints are respected with constraints (20) to (23). Constraints (20) and (21) respectively check that an inbound customer is picked up at home after their release date and dropped off at the depot before their due date, while constraints (22) and (23) check that an outbound customer is picked up at the depot after their release date and dropped off at home before their due date. Finally, constraints (24) to (27) define the variables.

4. Solution method

Solving the compact formulation (1)-(29) using a standard branch-and-cut solver is not efficient because of the weakness of the lower bound provided by the linear relaxation. In this section, we propose an extended formulation based on a set partitioning formulation (SP) which provides a much stronger lower bound. This formulation is characterized by a large number of decision variables. To cope with this drawback, we use a column generation approach to solve the linear relaxation. This linear relaxation of the extended formulation is called the master problem (MP). The master problem limited to the available set of columns is called the restricted master problem (RMP). After solving the RMP, a subproblem (called pricing problem) is solved in order to find new columns to add to the RMP. These columns correspond to routes with negative reduced costs. In our case, the pricing problem is a multi-trip elementary shortest path problem with resource constraints. It is solved using a specific label correcting algorithm (Feillet et al., 2004). The column generation algorithm ends when the pricing problem does not find any column with a negative reduced cost. Column generation is embedded into a branch-and-price algorithm. Due to the multi-trip aspect of the problem formulation, and because we later compare it with another implementation of branch-and-price, we call the algorithm the multi-trip branch-and-price algorithm (MT-BP).

4.1. Master problem

Let Ω be the set of feasible routes, as defined in Section 3.1. These routes satisfy capacity and time constraints. Customer locations are visited at most once. Let $a_i^r = 1$ if route $r \in \Omega$ serves request $i \in \mathcal{N}$, 0 otherwise. Let $b_{ij}^r = 1$ if route r uses arc $(v_i, v_j) \in \mathcal{A}$, 0 otherwise. We denote by $c_r = \sum_{(v_i,v_j)\in\mathcal{A}} b_{ij}^r c_{ij}$ the cost of route $r \in \Omega$. We introduce a non-negative integer decision variable θ_r equal to the number of times route $r \in \Omega$ is selected in the solution. The problem is described with the following set partitioning formulation (SP):

$$\min \sum_{r \in \Omega} c_r \theta_r \tag{30}$$

subject to

$$\sum_{r \in \Omega} a_i^r \theta_r = 1 \qquad \forall i \in \mathcal{N} \tag{31}$$

$$\sum_{r \in \Omega} \theta_r \leq K \tag{32}$$

$$\theta_r \in \mathbb{N} \qquad \forall r \in \Omega \tag{33}$$

The objective function (30) minimizes the total cost. Constraints (31) are the set partitioning constraints, they ensure that each request is served. Constraint (32) imposes to build at most K routes. The linear relaxation of (30)-(33) defines the master problem. We call $MP(\Omega_t)$, the master problem, restricted to the subset of routes $\Omega_t \subseteq \Omega$.

4.2. Column generation

Column generation consists in finding columns with negative reduced cost and adding them to the restricted master problem using dual variables of the current solution of the restricted master problem. Let $\forall v_i \in \mathcal{V} \backslash \{v_0\}$, λ_i unsigned be the dual variables associated with constraints (31) and $\lambda_0 \leq 0$ the dual variable associated with constraint (32). The algorithm starts with an initial set of feasible columns Ω_0 to solve the restricted master problem $MP(\Omega_0)$. Then, the pricing problem searches for columns with negative reduced costs and add them to the restricted master problem $MP(\Omega_1)$. This process is repeated until no column with a negative reduced cost is found. The reduced cost of route r is defined as follows:

$$\overline{c_r} = c_r - \left(\sum_{i \in \mathcal{N}} a_i^r \lambda_i \right) - \lambda_0 = \sum_{(v_i, v_j) \in \mathcal{A}, v_j \neq v_0} b_{ij}^r \left(c_{ij} - \lambda_j \right) + \left(\sum_{(v_i, v_0) \in \mathcal{A}} b_{i0}^r c_{i0} \right) - \lambda_0 \tag{34}$$

Note that dual variable λ_0 should not be subtracted every time the depot is visited.

Algorithm 1 describes the column generation scheme. MP(Ω_t) corresponds to solving the master problem restricted to Ω_t and is used to determine the solution S and the dual variable vector λ. Pricing(λ) corresponds to the call to the algorithm that solves the pricing problem with dual variables λ. It returns a set of routes with negative reduced costs. The pricing problem as well as an associated exact solution approach are described in detail in Section 5.

Algorithm 1 Column generation()

```
t ← 0
Ω_0 ← initial set of routes
repeat
    S, λ ← MP(Ω_t)
    R ← Pricing(λ)
    Ω_{t+1} ← Ω_t ∪ R
    t ← t + 1
until R = ∅
return S
```

4.3. Branching rules

Before describing the branching strategy, let us introduce some definitions. Given a solution S of $MP(\Omega_t)$, we call *flow* on arc $(v_i, v_j) \in \mathcal{A}$, the sum of variables using this arc in S, we denote it as $f_{ij} = \sum_{r\in\Omega} b^r_{ij}\theta_r$. A well-known branching strategy for VRPs consists in looking for an arc with a fractional flow and branch on it. When all arcs have an integer flow, the solution is feasible for the integer problem. In our problem, due to the use of multiple trips, it may happen that a solution is non-feasible for the integer problem but, has integer flows for all arcs. To deal with this issue, we introduce the notion of triplet. We call a *triplet* the succession of three vertices with v_0 as second vertex, *i.e.*, (v_i, v_0, v_j) is a triplet with $1 \leq i, j \leq n$. Triplet (v_i, v_0, v_j) can be seen as the succession of arcs (v_i, v_0) and (v_0, v_j); actually, it represents the transition between the ending part of a trip and the starting of the next trip in a route. Similarly to the definition of arc flow, the flow on a triplet (v_i, v_0, v_j) is defined as the sum of variables using this triplet in S, we denote it as $f_{i0j} = \sum_{r\in\Omega} b^r_{i0j}\theta_r$, where $b^r_{i0j} = 1$ if subsequence (v_i, v_0, v_j) exists in route r, 0 otherwise.

We show that any solution with integer flows both on arcs and on triplets is integer. We then explain how we use triplets for branching. But first, let us emphasize that a solution might have integer arc flows on all arcs and some fractional flows on triplets. We illustrate that with an example. Let us consider a solution of the master problem defined as a set of 4 routes covering 5 requests: $r_1 = (v_0\ v_1\ v_0\ v_2\ v_0)$ with $\theta_1 = 0.5$, $r_2 = (v_0\ v_1\ v_0\ v_3\ v_0\ v_4\ v_5\ v_0)$ with $\theta_2 = 0.5$, $r_3 = (v_0\ v_4\ v_5\ v_0)$ with $\theta_3 = 0.5$, and $r_4 = (v_0\ v_2\ v_0\ v_3\ v_0)$ with $\theta_4 = 0.5$. In this solution, $f_{ij} \in \{0, 1\}$ on all arcs $(v_i, v_j) \in \mathcal{A}$, but triplet (v_1, v_0, v_2), for example, has a fractional flow $f_{102} = 0.5$ (supplied by route r_1).

Proposition 1. *Let S be an optimal solution of MP. S is integer if and only if flows on all arcs and triplets are integer.*

Proof. The fact that all flows on arcs and triplets are integer when a solution is integer is trivial because flows are sums of variables θ_r. We show that having integer flows on arcs and triplets ensures that the solution is integer.

We consider a solution S that is not integer. We show that there exists at least one arc or one triplet for which the flow is fractional. We proceed by contradiction, assuming that the flow is integer for all arcs and triplets. Let r_1 be a route such that $0 < \theta_{r_1} < 1$ and let (v_0, v_u) be the first arc of this route. Let r_2 be another route in the solution (*i.e.*, $\theta_{r_2} > 0$) that also traverses arc (v_0, v_u). Route r_2 exists because the flow on arc (v_0, v_u) is integer and $0 < \theta_{r_1} < 1$.

We first assume that route r_2 does not start with (v_0, v_u) and call (v_l, v_0) the arc that precedes (v_0, v_u) in r_2. We know that $f_{l0u} = \sum_{r\in\Omega} b^r_{l0u}\theta_r \geq \theta_{r_2} > 0$ and also that f_{l0u} is integer. Then, $f_{l0u} + \theta_{r_1} > 1$, while $f_{l0u} + \theta_{r_1} \leq \sum_{r\in\Omega} b^r_{0u}\theta_r \leq \sum_{r\in\Omega} a^r_u\theta_r = 1$, which makes a contradiction.

We now consider the case when r_2 starts with the same arc as r_1 (v_0, v_u). Let then call v_i the first node after which r_1 and r_2 diverge. Node v_i exists because the two routes start with the same arc and are different. We consider two cases:

- $v_i \in \mathcal{V} \setminus \{v_0\}$. We denote v_{j_1} the node that follows v_i in route r_1 and v_{j_2} the node that follows v_i in route r_2. We know that: $\sum_{r\in\Omega} a^r_i\theta_r = 1$, $\sum_{r\in\Omega} a^r_i\theta_r \geq \sum_{r\in\Omega}(b^r_{ij_1} + b^r_{ij_2})\theta_r = f_{ij_1} + f_{ij_2}$, $f_{ij_1} > 0$ and $f_{ij_2} > 0$. It follows $0 < f_{ij_1} < 1$ and $0 < f_{ij_2} < 1$, which contradicts the initial assumption.

- $v_i = v_0$. We know, by definition, that v_i is not the first vertex of the two routes. We also know that it is not the last for at least one route, say route r_2, because the two routes are

different. So, v_i corresponds to an intermediate stop at the depot for route r_2. We call v_l the customer before v_i in the two routes and v_u the customer after v_i in route r_2. We known that $f_{l0u} = \sum_{r\in\Omega} b^r_{l0u}\theta_r \geq \theta_{r_2} > 0$ and also that f_{l0u} is integer. Then, $f_{l0u} + \theta_{r_1} > 1$, while $f_{l0u} + \theta_{r_1} \leq \sum_{r\in\Omega} b^r_{l0}\theta_r \leq \sum_{r\in\Omega} a^r_l\theta_r = 1$, which makes a contradiction.

□

Based on Proposition 1, we propose the following branching scheme. If S is not integer and there exists at least one arc (v_i, v_j) with a fractional flow f_{ij}, then we derive two branches: in the first branch, arc (v_i, v_j) is forbidden in the solution ($f_{ij} = 0$); in the second branch, arc (v_i, v_j) is enforced ($f_{ij} = 1$). To forbid the use of (v_i, v_j), variables using this arc are removed from the master problem and the arc is removed in the pricing problem. To enforce the use of (v_i, v_j), variables using arcs (v_i, v_l) with $v_l \neq v_j$ and (v_l, v_j) with $v_l \neq v_i$ are removed from the master problem and these arcs are removed in the pricing problem.

If a solution S is not integer but all arcs have an integer flow, then there exists a fractional flow on a triplet (Proposition 1). Let (v_i, v_0, v_j) be such a triplet. We derive two branches: in the first branch, the triplet (v_i, v_0, v_j) is forbidden in the solution ($f_{i0j} = 0$); in the second branch, the triplet (v_i, v_0, v_j) is enforced ($f_{i0j} = 1$). To forbid the use of (v_i, v_0, v_j), variables using it are removed from the master problem. In the pricing problem (see Section 5), a label that finishes with (v_i, v_0) is never extended to v_j. This constraint has some consequences on the dominance rule as explained in Section 5.4. To enforce the use of (v_i, v_0, v_j), variables using arcs (v_i, v_l) with $v_l \neq v_0$ and (v_l, v_j) with $v_l \neq v_0$ are removed from the master problem. Also, variables using arc (v_i, v_0) not followed by (v_0, v_j) and variables using arc (v_0, v_j) not preceded by (v_i, v_0) are removed from the master problem. In the pricing problem, arcs (v_i, v_l) with $v_l \neq v_0$ and (v_l, v_j) with $v_l \neq v_0$ are removed. Also, during the execution of the labeling algorithm, all triplets (v_i, v_0, v_l) with $v_l \neq v_j$ and (v_l, v_0, v_j) with $v_l \neq v_i$ are forbidden. To forbid a triplet, a label that finishes with the first arc of the triplet cannot be extended using the second arc. Again, it impacts the dominance rule as explained in Section 5.4.

4.4. Management of infeasible linear programs

At each node of the branch-and-price tree, it may happen that the current set of columns does not allow finding a feasible solution in the RMP. To deal with that issue, we introduce a non-negative decision variable z with a coefficient 1 in the left-hand side of Constraints (31). This variable is penalized in the objective function with a large coefficient M_z. With this variable a feasible solution always exists. When a LP is solved, variable z might then be equal to 0 or be positive. If $z = 0$, the algorithm continues normally. If $z > 0$, it is probable that the RMP does not admit any feasible solution (in the sense of the formulation without z). However, we cannot be sure: it might happen that a solution with $z = 0$ exists but that this solution is not as good as the solution with $z > 0$. We propose to deal with that as follows.

At the root node, M_z is set to a large initial value M^0_z. When the column generation at a given node N terminates with $z > 0$, the node is pruned, but a duplicate node N' is inserted in the pending-nodes queue, with exactly the same characteristics except that $M'_z = \alpha_z M_z$, with $\alpha_z > 1$. With this mechanism, the LP bound will progressively increase when nodes are duplicated, until $z = 0$ or a large-enough value is reached proving that the instance is not feasible. For each instance we compute a feasibility bound FB, if no feasible solution is found and the pending nodes all have a lower bound greater than FB, it proves that the instance is unfeasible.

5. Multi-Trip Shortest Path Problem with Resource Constraints

The pricing problem for vehicle routing problems is classically reduced to an Elementary Shortest Path Problem with Resource Constraints (ESPPRC), with ad hoc resource definitions. The ESPPRC is then solved using a labeling algorithm (see for example Feillet et al. (2004)). A labeling algorithm consists in using labels to represent partial routes. Partial routes are obtained by extending initial labels from a source node (the depot) in order to reach a final node (the depot). The labels are obtained using an extension function from one node to its neighbors. The goal is to find feasible routes with negative reduced costs. A dominance rule is used in order to eliminate partial routes that cannot create an optimal route.

Different resource definitions have been proposed in the context of pickup and delivery problems, but these definitions are always supported by a graph in which two nodes are introduced for each request: a pickup node and a drop-off node. When one of these nodes is traversed, a specific discrete resource is updated, to indicate that the associated request is either started or finished. In the MT-BP, we propose a completely different approach. In the graph, we only introduce a node for customer home locations, in addition to the depot. It drastically reduces the size of the graph but induces a complex resource management because we cannot identify easily in a label which requests are ongoing.

The main issues can be summarized as follows. When a label reaches a customer location v_i, it means that the associated request is added to the trip. If the request is inbound, it doesn't raise any special difficulty: the new patient is loaded into the vehicle, the time is updated to take account of the release date and a new deadline is defined to make sure that the due date will be satisfied. Outbound requests are much more difficult to manage. If v_i is the drop-off point of an outbound request, it reveals that the patient is in the vehicle from the beginning of the trip. We introduce a dedicated resource $\tilde{q}$ to manage that issue and efficiently check that adding this patient complies with the vehicle capacity. Furthermore, the starting time of the trip is potentially delayed because of the release date r_i (at which the patient was ready at the depot). In order to manage that second difficulty, we anticipate delays: without deciding which outbound requests will be added to the trip, we generate labels that leave the depot with delayed departures. Then, further delays are not allowed when extending labels: if the starting time of the trip is too early for an outbound request, this request can't be added to the trip. The information on trip starting times is not kept with a new resource: the vector of reachable requests (classically introduced to avoid that the same request is satisfied several times in a route) allows keeping the information on acceptable outbound requests. This vector is, however, somehow difficult to manage because a request could be unreachable in the current trip but could become reachable in the next trip.

In the following we introduce the main components of the labeling algorithm: label definition, label initialization, extension rules and dominance rules.

5.1. Label definition

A label L represents a path P starting at depot v_0 in graph $\mathcal{G}$. It is defined by a vector $L = (v, c, t, l, q, \tilde{q}, R = \{R_1, \ldots, R_n\})$ of size $n + 6$, with:

- v: the last node visited in P.
- c: the current reduced cost of P.
- t: the time at which node v is left (service time at node v included).
- l: the latest time allowed to be back to the depot in the current trip.

- q: the load induced by inbound requests in the current trip (including node v).
- $\tilde{q}$: the maximal load reached up to node v (included) in the current trip.
- R: the reachability vector.

Resource l is introduced to limit the arrival time at the depot and ensure satisfaction of the due date for inbound requests served in the trip. Resources q and $\tilde{q}$ allow managing the capacity constraint in presence of inbound and outbound requests. Resource $\tilde{q}$ is actually enough to check the capacity constraint, but q is needed to update $\tilde{q}$. Resource vector R is defined with three possible states for each request. Given a request $i \in \mathcal{N}$: (i) $R_i = C$ if v_i is reachable in the current trip (ii) $R_i = N$ if v_i is unreachable in the current trip because of resource constraints but is reachable in the next trip (iii) $R_i = U$ if node v_i cannot be reached anymore in the route because it has already been reached or because resource constraints make it unreachable. Other fields in L are standard.

5.2. Label initialization

Usually, labeling algorithms are initialized with a single label that corresponds to the path $P = (v_0)$, ready for extension at time 0. In the MT-BP, we propose to initialize the algorithms with $|\mathcal{N}^-| + 1$ labels:

- $L_0 = (v_0, -\lambda_0, 0, T_{max}, 0, 0, R)$ with:
  - $\forall k \in \mathcal{N}^+$: if $\max(r_k, t + t_{0k}) + s_k + t_{k0} \leq d_k$, $R_k = C$; $R_k = U$ otherwise
  - $\forall k \in \mathcal{N}^-$, $R_k = N$
- $L_i = (v_0, -\lambda_0, r_i + T_P, T_{max}, 0, 0, R)$ for $i \in \mathcal{N}^-$ with:
  - $\forall k \in \mathcal{N}^+$: if $\max(r_k, t + t_{0k}) + s_k + t_{k0} \leq d_k$, $R_k = C$; $R_k = U$ otherwise
  - $\forall k \in \mathcal{N}^-$: if $r_k \leq r_i$ and $t + t_{0k} \leq d_k$, $R_k = C$; if $r_k > r_i$ and $t + t_{0k} \leq d_k$, $R_k = N$; $R_k = U$ otherwise

Label L_0 represents a vehicle that would start empty from the depot at time 0. Pickup time T_P is not consumed before departure. Therefore, this vehicle can only serve inbound requests in this trip. Other labels correspond to later departures from the depot. We create one label for each time instant $r_i + T_P$ (label L_i), with $i \in \mathcal{N}^-$. These labels simulate vehicles that would start at time $r_i + T_P$. Patients of outbound requests with a release date not larger than r_i are apt to be in the vehicle, but none is necessarily in. No other labels are needed because there is no interest to start a route at any other time instant.

Resource R indicates which customers might be reached in the trip or in next trips. This condition depends on the deadline d_k for outbound customers. For inbound customers, $R_k = U$ if the deadline cannot be met, otherwise $R_k = C$ or $R_k = N$ depending on condition $r_k \leq r_i$.

5.3. Extension rules

In this section, we describe resource extension functions when a label is extended to a customer and when it is extended to the depot. These functions differ between inbound and outbound requests for some resources.

Extension to a customer

A label $L^1 = (v^1, c^1, t^1, l^1, q^1, \tilde{q}^1, R^1)$ attached to a node $v^1 = v_i \in \mathcal{V}$ is extended to a node $v_j \in \mathcal{V} \setminus \{v_0\}$ if and only if v_j is reachable in the current trip, *i.e.*, $R^1_j = C$. The new label L^2 is given by $L^2 = (v^2, c^2, t^2, l^2, q^2, \tilde{q}^2, R^2)$ such that:

- $v^2 \leftarrow v_j$, $c^2 \leftarrow c^1 + c_{ij} - \lambda_j$
- if $j \in \mathcal{N}^+$:
  - $t^2 \leftarrow \max(r_j, t^1 + t_{ij}) + s_j$
  - $l^2 \leftarrow \min(l^1, d_j)$
  - $q^2 \leftarrow q^1 + q_j$
  - $\tilde{q}^2 \leftarrow \max(\tilde{q}^1, q^2)$

  The service cannot start before the release date (resource t). The deadline is updated if needed (resource l). The load induced by inbound requests is increased (resource q). Resource $\tilde{q}$ is increased if the load q^2 at node v is larger than the highest load up to that node.
- if $j \in \mathcal{N}^-$:
  - $t^2 \leftarrow t^1 + t_{ij} + s_j$
  - $l^2 \leftarrow l^1$
  - $q^2 \leftarrow q^1$
  - $\tilde{q}^2 \leftarrow \tilde{q}^1 + q_j$

  Resource t is simply updated. Neither the deadline (resource l) nor resource q are modified. Resource $\tilde{q}$ is increased by q_j because visiting v_j reveals that the patient was in the vehicle from the beginning of the trip.
- $R^2_j \leftarrow U$
- $\forall k \in \mathcal{N}^+ \setminus \{j\}$:
  - if $R^1_k = C$ and $\max(r_k, t^2 + t_{jk}) + s_k + t_{k0} \leq \min(l^2, d_k)$, then $R^2_k \leftarrow C$
  - else if $R^1_k \in \{C, N\}$, $q^2 = 0$ and $\max(r_k, t^2 + t_{j0} + t_{0k}) + s_k + t_{k0} \leq d_k$, then $R^2_k \leftarrow N$
  - else if $R^1_k \in \{C, N\}$, $q^2 > 0$ and $\max(r_k, t^2 + t_{j0} + T_D + t_{0k}) + s_k + t_{k0} \leq d_k$, then $R^2_k \leftarrow N$
  - else $R^2_k \leftarrow U$

  These conditions straightforwardly take account of the waiting times implied by release dates and the deadlines derived from due dates. Time T_D at the depot is introduced or not depending on the presence of inbound customers in the trip.
- $\forall k \in \mathcal{N}^- \setminus \{j\}$:
  - If $R^1_k = C$ and $t^2 + t_{jk} \leq d_k$ and $t^2 + t_{jk} + s_k + t_{k0} \leq l^2$, then $R^2_k \leftarrow C$
  - else if $R^1_k \in \{C, N\}$ and $q^2 = 0$ and $\max(r_k, t^2 + t_{j0}) + T_P + t_{0k} \leq d_k$, then $R^2_k \leftarrow N$
  - else if $R^1_k \in \{C, N\}$ and $q^2 > 0$ and $\max(r_k, t^2 + t_{j0} + T_D) + T_P + t_{0k} \leq d_k$, then $R^2_k \leftarrow N$
  - else $R^2_k \leftarrow U$

Computationally, the feasibility of label L^2 is checked before computing vector R^2. The only condition to check is $\tilde{q} \leq Q$. Time constraints are necessarily satisfied because the extension was conditioned by $R^1_j = C$.

Extension to the depot

Note that contrary to classical implementations of the ESPPRC for vehicle routing problems, labels that have been extended to the depot are still eligible for future extensions. Furthermore, as for the label initialization, several new labels are created, with different starting times. These labels represent the different times at which the vehicle might start its new trip.

A label $L^1 = (v^1, c^1, t^1, l^1, q^1, \tilde{q}^1, R^1)$ attached to a node $v^1 = v_i \in \mathcal{V} \setminus \{v_0\}$ can always be extended to the depot. Indeed, deadlines are checked in L_1 thanks to resources l and R_i (the label could not have reached v_i if a direct return to the depot was not possible). Such an extension means that the current trip is finished and that a new trip can start. We generate one label L_0^2 for which the new trip will start as early as possible, with the vehicle empty, and one label L_j^2 for each departure time $r_j + T_p$ obtained from requests $j \in \mathcal{N}^-$ such that $R_j^1 \in \{C, N\}$.

Label $L_0^2 = (v_0, c^1 + c_{i0}, a_t, T_{max}, 0, 0, R^2)$ where:

- a_t indicates when the vehicle is ready: $a_t = t^1 + t_{i0} + T_D$ if $q^1 > 0$ (a drop-off time is needed for inbound requests), $a_t = t^1 + t_{i0}$ otherwise
- $\forall k \in \mathcal{N}^+$:
  - if $R_k^1 \in \{C, N\}$ and $\max(r_k, a_t + t_{0k}) + s_k + t_{k0} \leq d_k$, then $R_k^2 \leftarrow C$
  - else $R_k^2 \leftarrow U$
- $\forall k \in \mathcal{N}^-$:
  - if $R_k^1 \in \{C, N\}$, then $R_k^2 \leftarrow N$
  - else $R_k^2 \leftarrow U$

Label $L_j^2 = (v_0, c^1 + c_{i0}, t^2 = \max(r_j, a_t) + T_P, T_{max}, 0, 0, R^2)$ where:

- a_t indicates when the vehicle is ready and is defined as above
- $\forall k \in \mathcal{N}^+$:
  - if $R_k^1 \in \{C, N\}$ and $\max(r_k, t^2 + t_{0k}) + s_k + t_{k0} \leq d_k$, then $R_k^2 \leftarrow C$
  - else $R_k^2 \leftarrow U$
- $\forall k \in \mathcal{N}^-$:
  - if $R_k^1 \in \{C, N\}$, and $t^2 + t_{0k} \leq d_k$, and $r_k \leq \max(r_j, a_t)$, then $R_k^2 \leftarrow C$
  - else if $R_k^1 \in \{C, N\}$, and $r_k > \max(r_j, a_t)$, then $R_k^2 \leftarrow N$
  - else $R_k^2 \leftarrow U$

5.4. Dominance rule

We introduce the following dominance rule in order to improve the efficiency of the algorithm. By definition, given two labels L^1 and L^2 attached to the same node, L^1 dominates L^2 if, (i) L^1 has a better cost and (ii) any feasible extension of L^2 is also feasible for L^1 (*i.e.*, each resource is less consumed by L^1 than L^2 and, each node is in a better (or equivalent) reachability state in L^1 than in L^2). In the following, the label identifier is added as a superscript to the parameters of the labels. Formally speaking, L^1 dominates L^2 if:

- $v^1 = v^2$, $c^1 \leq c^2$, $t^1 \leq t^2$, $l^1 \geq l^2$, $q^1 \leq q^2$, $\tilde{q}^1 \leq \tilde{q}^2$,
- for every $i \in \mathcal{N}$: if $R_i^2 = C$, then $R_i^1 = C$ and if $R_i^2 = N$, then $R_i^1 \in \{C, N\}$

Note that the dominance rule is not applied in a very specific situation, due to the fact that branching constraints on triplets are non-robust with regard to the pricing problem. Let us consider that a constraint on a triplet (v_i, v_0, v_j) is active at the current branch-and-price node. Let also consider two labels L_1 and L_2, both attached to the depot and such that L_1 originates from v_i. Then, L_1 is subject to a constraint that does not apply to L_2 and is thus more restricted in its extensions: L_1 cannot dominate L_2.

6. Computational experiments

In this section, we describe the benchmarks of instances used for our experiments and we analyze the computational results.

As mentioned in Section 2, the MTMPD-RD was not addressed in the literature. However, the MTMPD-RD has several similarities with the VRP with Simultaneous Pickup and Delivery and Time Windows (VRPSPDTW) for which benchmark instances are available (Wang and Chen, 2012). To allow a comparison on the basis of these instances, we adapted our branch-and-price algorithm to solve the VRPSPDTW.

We also generated a set of instances for the MTMPD-RD. In order to show the effectiveness of our algorithm (MT-BP), we developed a standard branch-and-price algorithm to solve the classical pickup and delivery problem. In the following we coin this algorithm PD-BP. In order to allow a comparison between MT-BP and PD-BP, the generated instances were slightly simplified.

All algorithms are implemented in the C++ programming language and Linear Programs are solved using the commercial solver IBM CPLEX 12.9. Experiments are executed on a computer with a 3.6 GHz CPU and 32 GB of RAM running on Windows 10. All computational times are given in seconds. Unless explicitly indicated, the computational time is limited to 7,200 seconds.

This section is organized in six parts. First, we explain the adaptation of our algorithm (MT-BP) to solve the VRPSPDTW and we evaluate it on the existing benchmark instances. Second, we explain how we generated benchmark instances for the MTMPD-RD. Third, we briefly describe the PD-BP algorithm and compare it to MT-BP on a slightly simplified version of the new benchmark instances. Fourth, we deeply analyze the behavior of MT-BP. Fifth, we study the impact of the size of the fleet of vehicles on the solution cost and on the computing time. Finally, we conduct other sensitivity analyses on challenging instances characterized by a larger interval between release and due dates and/or a larger time horizon.

6.1. Comparison with the VRPSPDTW literature

As mentioned previously, the VRPSPDTW has several similarities with the MTMPD-RD except for the few following points. In the VRPSPDTW, all customers require both a delivery (outbound) and a pickup (inbound) request. When visiting a customer location, a vehicle has to serve both requests, starting with the delivery request. Customer visits are subject to time windows at the customer location, instead of release dates and due dates for the MTMPD-RD. A service time is only considered at customer locations and includes both the time for delivery and for pickup. Finally, multiple trips are not allowed.

In the literature, the VRPSPDTW is usually defined as a lexicographic bi-objective problem. The first objective minimizes the number of vehicles, while the second objective minimizes the

total traveling cost. Readers are referred to Wang and Chen (2012) for a formal definition of the VRPSPDTW.

In order to solve the VRPSPDTW, we adapted the MT-BP algorithm as follows:

- In the pricing problem, we do not allow extending labels that have reached the depot. It prevents generating routes with multiple trips. When initializing labels and applying extension rules, all values N (reachable in next trip) of the reachability vector are marked as unreachable U in the reachability vector.
- The VRPSPDTW introduces a strict precedence between the inbound and outbound requests associated with the same customer. Recall that these two requests are considered independently in our model, *i.e.*, two nodes are introduced in $\mathcal{V}$ even if they correspond to the same physical location. To deal with the strict precedence between inbound and outbound requests, we define the inbound node of the customer location as the unique successor of the outbound node, and inversely, the outbound node as the unique predecessor of the inbound node. The distance and the travel time between these two nodes are fixed to zero.
- Time windows at the customer locations are converted into release and due dates as follows. Given a customer v_i in a VRPSPDTW instance with a time window $[a_i, b_i]$ and a service time s_i, we denote i^- (resp., i^+) the associated outbound (resp., inbound) request and we set $r_{i^-} = 0$, $d_{i^-} = b_i$, $s_{i^-} = 0$, $r_{i^+} = a_i$, $d_{i^+} = T_{max}$ and $s_{i^+} = s_i$. T_{max} is set to the closing time b_0 of the time window associated with the depot. T_P and T_D are set to 0.

For the sake of comparison, we set for each tested instances the number of vehicles equals to the number of vehicles obtained in the best known solution in the literature and our single objective is to minimize the total traveling distance.

To evaluate our algorithm, we use the benchmark instances of Wang and Chen (2012). They consist of three 10-customers instances, three 25-customers instances, three 50-customers instances and fifty-six 100-customers instances. Optimal solutions are only known for the three instances with 10 customers, for one instance with 25 customers and for one instance with 50 customers. These solutions were found by Wang and Chen using a commercial solver. Since 2012, several papers have addressed these instances with different heuristic solution methods. Shi et al. (2020) reports best-known solutions.

Table 3 shows our results on the nine small instances (10, 25 and 50 customers). Column "Instance" states the instance name; "#customers" the number of customers; "#vehicles" the number of vehicles used in the best known solution from the literature; "BKS" the value of the best known solution; "opt" the optimality status in the literature; "MT-BP" the value of the solution obtained by MT-BP, and "CPU time" the computing time in seconds of MT-BP.

Table 3 shows that, except for instance RCdp5004, we could solve all instances very quickly. Instance RCdp5004 could only be solved to optimality after more than 27 hours. We also conducted complementary experiments by imposing to use one less vehicle. We proved that the problem becomes infeasible in all cases (including instance RCdp5004). We thus closed the benchmark for small instances.

For instances with 100 customers, MT-BP was able to solve to optimality six instances out of the 56. Table 4 presents the results for these six instances. For these six instances, we found the

[1] A value of 725.59 was mistakenly reported in Wang and Chen (2012): the detail of the solution, provided by the authors, permitted to see that this solution is not feasible

Table 3: Comparison with the best-known solutions of the small-size instances for the VRPSPDTW

Instance	#customers	#vehicles	BKS	opt	MT-BP	CPU time
RCdp1001	10	3	348.98	yes	348.98	0.01
RCdp1004	10	2	216.69	yes	216.69	0.01
RCdp1007	10	2	310.81	yes	310.81	0.02
RCdp2501	25	5	551.05	yes	551.05	0.09
RCdp2504	25	4	473.46		473.46	0.25
RCdp2507	25	5	540.87		540.87	0.28
RCdp5001	50	9	994.18	yes	994.18	4.41
RCdp5004	50	6	733.21[1]		733.21	99234.80
RCdp5007	50	7	809.72		809.72	76.83

same value as the best-known solution in the literature; we also proved for four of them infeasibility when using one less vehicle. Thus we proved the optimality of four best-known solutions. When using one less vehicle, we were unable to solve Rdp106 and Cdp106 or to prove their infeasibility. For the fifty remaining instances, no feasible solution was found within two hours of computing time.

Table 4: Comparison with the best-known solutions of the 100-customers instances for the VRPSPDTW

Instance	#vehicles	BKS	MT-BP	CPU time
Rdp101	19	1650.80	1650.80	11.34
Rdp102	17	1486.12	1486.12	76.59
Rdp105	14	1377.11	1377.11	598.54
Rdp106	12	1252.03	1252.03	7117.61
Cdp105	10	1053.12	1053.12	257.114
Cdp106	10	963.45	963.45	4932.52

The benchmark instances of Wang and Chen (2012) are derived from the instances of Solomon (1987). Instances are classified according to two factors:

- Customer distribution: with clustered customers (C), with customer locations generated uniformly randomly over a square (R), with a combination of randomly placed and clustered customers (RC).
- Time window type and vehicle capacity: 1 for narrow time windows and small vehicle capacity, 2 for large time windows and large vehicle capacity.

Small-size instances of Wang and Chen (2012) are all of type RC1, which means a combination of randomly placed and clustered customers and, with narrow time windows and small vehicle capacity. 100-customers instances contain instances of any type (C1, R1, RC1, C2, R2, RC2) but MT-BP was able to solve only instances of type 1.

Even though we closed the benchmark for some instances, we can conclude that the MT-BP is not very suitable to address VRPSPDTW instances with 100 customers, as it did not find any feasible solution for most of such instances within the considered time limit. In particular, instances involving large time windows were harder to solve, as none of them could be solved. This behaviour is often

encountered when using column generation to solve vehicle routing problems. In fact, enlarging time windows increases the number of feasible solutions in the dynamic programming process when solving the subproblems, which is a key element in the effectiveness of the solution method. We also notice that MT-BP is able to solve instances of different types of customer distribution (randomly located (R) and clustered customers (C)).

6.2. Generation of instances for the MTMPD-RD

In this section, we describe how we generated a set of realistic instances for the MTMPD-RD. The instances were generated using real data from the city of Aix-en-Provence, France. The depot is the Public Hospital of the city and the demands are unitary: they correspond to the transportation of single patients from or to this hospital. The road network of the city is extracted from OpenStreetMap and represented by a graph denoted $\mathcal{G}_C$. The travel times are expressed in minutes and rounded up to integer values. The speeds considered are half of the maximal speeds indicated on each road segment in OpenStreetMap, in order to match with the urban context.

To create an instance with n requests, we randomly select n nodes from $\mathcal{G}_C$. From these n nodes plus the depot node, we compute the travel time matrix with the Dijkstra's shortest path algorithm minimizing traveling time and we construct the complete graph $\mathcal{G}$ and the input data t_{ij}. We set $c_{ij} = t_{ij}$. The time horizon is set to $[0\,min., 240\,min.]$ corresponding to a working period of 4 hours. Pickup and drop-off times at the depot (T_P and T_D) are fixed to 3 minutes, as well as service times at customer locations.

To generate release and due dates, we suppose that an acceptable ride time for a customer is the time needed for a direct ride between pickup and drop-off locations plus 20 minutes. To ensure feasibility for each request i, we randomly generate r_i and d_i as follows:

- if $i \in \mathcal{N}^+$, then $r_i \geq t_{0i}$, $d_i \leq T_{max} - T_D$, and $d_i - r_i = s_i + t_{i0} + 20$.
- if $i \in \mathcal{N}^-$, then $r_i \geq 0$, $d_i \leq T_{max} - s_i - t_{i0}$, and $d_i - r_i = T_P + t_{0i} + 20$.

Four series of instances are generated with 50 or 100 customers and two types of vehicles: sedan ($Q = 2$) and minivan ($Q = 4$). Series of instances are named $S_{n/t}$ where, n is the number of requests (50 or 100) and t is the vehicle type (s for sedan or m for minivan). The fleet size differs from an instance to another and is determined in a way that guarantees both feasibility and a high utilization rate of vehicles (see details below).

Given n and t, series $S_{n/t}$ is composed of 3 subsets of instances corresponding to different distributions of inbound and outbound requests. The probability that a request is inbound is fixed to: 25%, 50% or 75%. Each subset is composed of 5 instances identified from 1 to 5. We then obtain a total of 15 instances per series. Instances are named $I_{n/t/p/i(k)}$, with n the number of customers, t the vehicle type, p the probability that a request is inbound, i the instance id, and k the number of vehicles.

We first generated the set of instances $S_{50/s}$. This set of instances is used to generate $S_{50/m}$ by setting Q to 4. In both sets we determine for each instance I, the minimum number of needed vehicles to ensure feasibility. So, the size of the fleet of vehicles K can differ from one instance to another.

100-customers instances are generated by combining 50-customers instances. Given a vehicle type t and a probability p that a request is inbound, an instance $I_{100/t/p/i(k)}$ is a combination of instances $I_{50/t/p/i_1(k_1)}$ and $I_{50/t/p/i_2(k_2)}$ where $i = i_1$, $i_2 = i_1 \pmod 5 + 1$ and $k = k_1 + k_2$ to ensure feasibility.

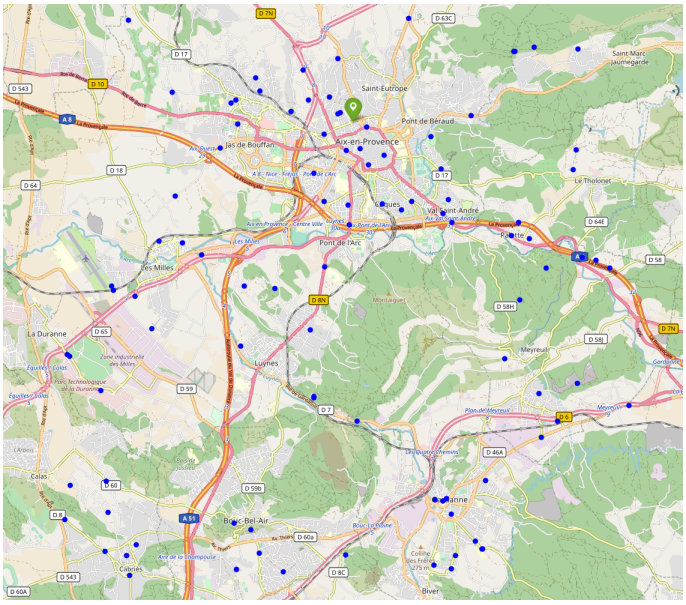

Figure 2: Customers distribution of instance $I_{100/s/25/5(15)}$

So, we generate 4 series of instances characterized by the number of requests and the type of vehicles: $S_{50/s}$, $S_{50/m}$, $S_{100/s}$ and $S_{100/m}$ with 15 instances each.

Even though patient locations are selected randomly and uniformly among the graph $\mathcal{G}_\mathcal{C}$, their distribution is not uniform in space. The graph $\mathcal{G}_\mathcal{C}$ represents a real road network and the denser an area is, the more nodes it contains. Consequently, nodes are more likely to be selected in urban areas than in the countryside. As an illustration, Figure 2 shows the customer distribution of instance $I_{100/s/25/5(15)}$ (and of instance $I_{100/m/25/5(15)}$ as only the vehicle capacity and fleet size change). The green flag corresponds to the Public Hospital of Aix-en-Provence, while the small blue points are customer locations. We can see that the distribution mixes clusters and random nodes. The biggest cluster is in the city of Aix-en-Provence, other smaller clusters exist in smallest cities like Gardanne or Bouc-Bel-Air, but also some nodes are more isolated.

6.3. Comparison against a pickup and delivery branch-and-price algorithm

As mentioned previously, the MTMPD-RD problem is formulated as a multi-trip vehicle routing problem but it can also be seen as a Pickup and Delivery Problem, with the pickup (resp., delivery) node as the depot of outbound (resp., inbound) requests. In order to transform the MTMPD-RD into a pickup and delivery problem, we duplicate the depot for each request, giving rise to a graph with $2n + 1$ nodes. In order to convert the MTMPD-RD problem into a Pickup and Delivery Problem, we need to simplify the MTMPD-RD by removing the service time at the depot location (*i.e.*, $T_D = T_P = 0$). Indeed, the service time is associated with the depot and cannot be duplicated for each equivalent node when moving to the Pickup and Delivery formulation. With this representation, the MTMPD-RD problem can be solved using a standard branch-and-price for pickup and delivery problems.

To solve the problem, we implemented a branch-and-price algorithm based on Garaix et al. (2010). This algorithm is denoted PD-BP for Pickup and Delivery Branch-and-Price. For a fair comparison, the symmetries induced by the $n + 1$ copies of the depot is broken by removing arcs (v_i, v_j) with $i > j$ where v_i and v_j correspond to the depot.

Tables 5 and 6 provide experimental results on the generated instances with respectively 50 customers and 100 customers (recall that in the experiments of this section the service time at the depot location is ignored). They show for both solution approaches, the value of the lower bound at the root node ("RLB"), the computing time at the root node ("Time^{RLB}") in seconds, the objective value of the best integer solution found at the end of the branch-and-price algorithm ("UB") and the total computing time ("Time") in seconds. For "RLB" and "UB", the symbol "-" means that no solution has been found within the allocated computing time. For the computing time, the symbol "-" means that the time limit of two hours was reached. Upper bounds are given in bold when optimality is proven.

The two tables show that MT-BP significantly outperforms PD-BP. Both algorithms found the root lower bound for all instances but, the running time is most of the time better for the MT-BP (for 59 instances out of 60). One might note that this difference grows significantly for the 100-customers instances. This shows that the column generation of MT-BP is much more efficient than the one of PD-BP. PD-BP found a feasible solution for all 50-customers instances and proved optimality for 26 of them, while MT-BP proved optimality for all instances. For the 100-customers instances, PD-BP found a feasible solution for 13 out of 30 instances and proved optimality for 4 of them, while MT-BP found a feasible solution for 22 out of 30 instances and proved optimality for 16 of them. Those figures highlight the efficiency of MT-BP against PD-BP.

6.4. Detailed analysis of MT-BP

In this section, we conduct a deep analysis on the behaviour of MT-BP. Tables 7 and 8 provide results of the MT-BP algorithm on the benchmark instances that we generated. Note that contrary to the instances used in Section 6.3, service times T_P and T_D are not ignored. These tables show for each instance, the value of the lower bound ("RLB") at the root node, the computing time ("Time^{RLB}") in seconds to reach "RLB", the objective function value of the best integer solution found ("UB") and the total computing time ("Time") in seconds of MT-BP. The symbol "-" for "Time^{RLB}" and "Time" means that the process did not finish within the time limit of two hours. For the other parameters, the symbol "-" means that no value was found within the time limit of two hours.

UB is highlighted in bold if optimality is proven. We also provide the gap at the root node ("root gap") between RLB and the optimal solution and, the final gap ("final gap") when MT-BP terminates with a non-proven optimum. The value "root gap" is only provided if optimality is proven. Columns "#trips", "min", and "max" respectively provide the total number of trips, the minimum number of trips per vehicle, and the maximum number of trips per vehicle in the obtained solution.

From the two tables, we can notice that the root lower bound is found for all instances and the gap at the root node is very small ; it rarely exceeds 2% and never exceeds 3% (except for instance $I_{50/m/25/4(7)}$ with a gap of 5.09%) which denotes the high quality of the lower bound provided by the set partitioning formulation and the efficiency of the column generation algorithm. When a feasible solution is found but optimality not proven, the final gap is lower than 1% which also shows the high quality of the solutions found within two hours of computing time.

All 50-customers instances are solved optimally and, mostly with a computing time of less than 10 seconds. For 100-customers instances, 21 feasible solutions are found and optimality is proven for 18 of these instances. These figures show that MT-BP is still efficient on instances with up to 100 customers. One might notice that the algorithm is more efficient with minivans (13 feasible solutions with 11 optimal solutions) than sedans (8 feasible solutions with 7 optimal solutions). As we can notice from the last three columns of Table 8, using sedan vehicles requires more trips

Table 5: Comparison of the PD-BP and the MT-BP on 50-customers instances

	PD-BP				MT-BP			
Instance	RLB	Time^{RLB}	UB	Time	RLB	Time^{RLB}	UB	Time
$I_{50/s/25/1(8)}$	1079.50	1.6	**1092**	4.9	1079.50	1.8	**1092**	8.3
$I_{50/s/25/2(7)}$	933.19	4.6	**934**	5.0	933.19	2.4	**934**	3.5
$I_{50/s/25/3(7)}$	957.00	3.9	**959**	12.1	957.00	2.7	**959**	11.5
$I_{50/s/25/4(8)}$	903.00	3.0	**911**	155.4	903.00	2.3	**911**	8.4
$I_{50/s/25/5(7)}$	859.50	3.1	**865**	9.7	859.50	2.7	**865**	9.4
$I_{50/s/50/1(7)}$	882.33	2.5	886	-	882.33	1.2	**886**	6.1
$I_{50/s/50/2(8)}$	769.79	4.2	**771**	4.5	769.79	0.9	**771**	1.3
$I_{50/s/50/3(9)}$	996.50	1.4	**997**	1.6	996.50	1.0	**997**	1.3
$I_{50/s/50/4(7)}$	917.50	2.1	**928**	3.9	917.50	1.7	**928**	3.2
$I_{50/s/50/5(7)}$	849.50	2.7	**855**	3.5	849.50	1.7	**855**	3.4
$I_{50/s/75/1(7)}$	942.25	1.7	**948**	2.7	942.25	1.1	**948**	2.5
$I_{50/s/75/2(7)}$	878.00	2.8	**878**	2.8	878.00	1.0	**878**	1.0
$I_{50/s/75/3(7)}$	903.00	1.6	**913**	10.3	903.00	1.2	**913**	11.9
$I_{50/s/75/4(7)}$	922.00	1.6	**933**	8.5	922.00	0.7	**933**	4.8
$I_{50/s/75/5(8)}$	995.00	4.0	999	-	995.00	0.6	**999**	1.5
$I_{50/m/25/1(8)}$	1021.50	1.8	**1031**	3.9	1021.50	1.6	**1031**	3.4
$I_{50/m/25/2(7)}$	824.27	5.9	**829**	10.0	824.27	2.7	**829**	9.7
$I_{50/m/25/3(7)}$	906.50	7.2	**914**	47.9	906.50	2.8	**914**	8.7
$I_{50/m/25/4(7)}$	874.50	4.7	**886**	25.7	874.50	3.3	**886**	82.0
$I_{50/m/25/5(7)}$	811.00	4.5	**811**	4.5	811.00	2.8	**811**	3.3
$I_{50/m/50/1(7)}$	868.50	3.1	**875**	12.8	868.50	1.2	**875**	7.9
$I_{50/m/50/2(7)}$	736.50	11.9	**742**	41.8	736.50	1.6	**742**	4.5
$I_{50/m/50/3(9)}$	969.00	3.1	**974**	4.4	969.00	1.0	**974**	1.9
$I_{50/m/50/4(7)}$	902.54	2.9	**913**	4.5	902.54	1.9	**913**	3.1
$I_{50/m/50/5(7)}$	774.00	3.7	**779**	5.1	774.00	1.9	**779**	3.3
$I_{50/m/75/1(6)}$	888.40	14.0	**897**	32.6	888.40	1.7	**897**	3.0
$I_{50/m/75/2(7)}$	840.00	5.2	**842**	196.5	840.00	0.9	**842**	1.4
$I_{50/m/75/3(7)}$	862.50	4.0	867	-	862.50	1.2	**867**	2.9
$I_{50/m/75/4(7)}$	863.00	3.5	**871**	342.9	863.00	0.7	**871**	4.4
$I_{50/m/75/5(7)}$	931.83	15.3	951	-	931.83	0.7	**949**	2.9

Table 6: Comparison of the PD-BP and the MT-BP on 100-customers instances

	PD-BP				MT-BP			
Instance	RLB	Time^{RLB}	UB	Time	RLB	Time^{RLB}	UB	Time
$I_{100/s/25/1(15)}$	1823.50	94.7	-	-	1823.50	63.0	-	-
$I_{100/s/25/2(14)}$	1710.75	258.8	-	-	1710.75	133.1	1721	-
$I_{100/s/25/3(15)}$	1643.00	176.6	1655	-	1643.00	107.6	-	-
$I_{100/s/25/4(15)}$	1557.25	142.0	**1559**	151.0	1557.25	97.5	**1559**	152.0
$I_{100/s/25/5(15)}$	1710.00	88.5	1734	-	1710.00	67.1	1736	-
$I_{100/s/50/1(15)}$	1400.50	139.8	-	-	1400.50	32.2	-	-
$I_{100/s/50/2(17)}$	1546.00	39.7	1564	-	1546.00	34.3	1566	-
$I_{100/s/50/3(16)}$	1687.83	64.2	1692	-	1687.83	44.6	**1692**	1822.4
$I_{100/s/50/4(14)}$	1552.83	95.1	-	-	1552.83	53.6	1575	-
$I_{100/s/50/5(14)}$	1510.53	89.4	-	-	1510.53	39.8	**1519**	535.1
$I_{100/s/75/1(14)}$	1673.00	74.0	**1675**	77.7	1673.00	26.0	**1675**	35.4
$I_{100/s/75/2(14)}$	1581.33	156.3	-	-	1581.33	32.2	**1589**	919.7
$I_{100/s/75/3(14)}$	1632.00	108.4	-	-	1632.00	24.5	-	-
$I_{100/s/75/4(15)}$	1750.50	72.5	**1759**	356.1	1750.50	16.1	**1759**	58.1
$I_{100/s/75/5(15)}$	1756.50	89.5	-	-	1756.50	23.2	-	-
$I_{100/m/25/1(15)}$	1560.00	302.6	-	-	1560.00	60.8	**1573**	651.3
$I_{100/m/25/2(14)}$	1471.97	419.5	-	-	1471.97	98.1	1480	-
$I_{100/m/25/3(14)}$	1481.92	387.7	1491	-	1481.92	102.3	**1490**	1758.4
$I_{100/m/25/4(14)}$	1422.67	232.2	**1426**	263.9	1422.67	102.1	**1426**	244.3
$I_{100/m/25/5(15)}$	1525.43	319.0	-	-	1525.43	55.7	-	-
$I_{100/m/50/1(14)}$	1291.75	545.9	1309	-	1291.75	44.5	1306	-
$I_{100/m/50/2(16)}$	1401.46	312.3	-	-	1401.46	39.8	-	-
$I_{100/m/50/3(16)}$	1559.43	402.2	-	-	1559.43	39.0	**1575**	1612.5
$I_{100/m/50/4(14)}$	1445.94	168.1	1456	-	1445.94	52.9	**1456**	216.6
$I_{100/m/50/5(14)}$	1419.43	300.4	1425	-	1419.43	45.3	**1425**	79.9
$I_{100/m/75/1(13)}$	1483.50	481.6	-	-	1483.50	24.4	**1497**	502.0
$I_{100/m/75/2(14)}$	1458.20	548.3	-	-	1458.20	29.4	**1471**	515.5
$I_{100/m/75/3(14)}$	1486.50	211.9	1491	-	1486.50	26.2	**1491**	53.2
$I_{100/m/75/4(14)}$	1547.33	625.0	-	-	1547.33	18.0	**1554**	117.9
$I_{100/m/75/5(13)}$	1477.00	590.9	-	-	1477.00	24.6	-	-

than using minivans because of the smaller vehicle capacity. It may explain why the algorithm is more efficient with minivans. From these instances, the impact of the distribution of inbound and outbound requests does not seem to have an impact on the computing time or the quality of obtained solutions.

Table 7: Results of the MT-BP algorithm on 50-customers instances

	RLB		UB		Gaps (%)		Trips		
Instance	RLB	Time^{RLB}	UB	Time	root gap	final gap	nb trips	min	max
$I_{50/s/25/1(8)}$	1129.53	1.2	**1146**	6.6	1.44	0.00	23	2	4
$I_{50/s/25/2(7)}$	976.17	2.5	**988**	7.3	1.20	0.00	21	2	5
$I_{50/s/25/3(7)}$	1027.78	2.8	**1054**	21.6	2.49	0.00	23	3	4
$I_{50/s/25/4(8)}$	949.75	2.0	**956**	9.4	0.65	0.00	24	2	4
$I_{50/s/25/5(7)}$	889.83	2.7	**902**	16.1	1.35	0.00	23	2	5
$I_{50/s/50/1(7)}$	903.50	1.4	**910**	10.3	0.71	0.00	21	2	4
$I_{50/s/50/2(8)}$	771.00	0.9	**771**	0.9	0.00	0.00	18	1	4
$I_{50/s/50/3(9)}$	1036.00	1.0	**1047**	2.6	1.05	0.00	22	1	4
$I_{50/s/50/4(7)}$	964.00	1.5	**976**	5.6	1.23	0.00	20	2	4
$I_{50/s/50/5(7)}$	881.50	1.8	**884**	2.7	0.28	0.00	21	2	4
$I_{50/s/75/1(7)}$	948.00	1.1	**952**	2.0	0.42	0.00	23	2	4
$I_{50/s/75/2(7)}$	879.00	1.1	**879**	1.2	0.00	0.00	19	2	4
$I_{50/s/75/3(7)}$	905.50	1.4	**915**	9.3	1.04	0.00	23	2	5
$I_{50/s/75/4(7)}$	943.71	0.8	**967**	21.2	2.41	0.00	22	2	5
$I_{50/s/75/5(8)}$	1026.63	0.8	**1040**	11.6	1.29	0.00	24	2	4
$I_{50/m/25/1(8)}$	1065.00	1.5	**1069**	2.2	0.37	0.00	19	2	4
$I_{50/m/25/2(7)}$	892.50	2.1	**908**	21.7	1.71	0.00	18	2	4
$I_{50/m/25/3(7)}$	968.38	2.7	**993**	45.4	2.48	0.00	21	2	4
$I_{50/m/25/4(7)}$	962.43	3.0	**1014**	4.5	5.09	0.00	23	2	4
$I_{50/m/25/5(7)}$	857.02	2.6	**864**	6.0	0.81	0.00	19	2	3
$I_{50/m/50/1(7)}$	874.00	1.2	**877**	1.8	0.34	0.00	20	2	4
$I_{50/m/50/2(7)}$	743.33	1.4	**745**	2.8	0.22	0.00	16	1	3
$I_{50/m/50/3(9)}$	1015.00	0.8	**1026**	1.7	1.07	0.00	19	2	3
$I_{50/m/50/4(7)}$	949.00	1.6	**962**	6.1	1.35	0.00	19	2	4
$I_{50/m/50/5(7)}$	809.00	1.6	**809**	1.6	0.00	0.00	18	2	3
$I_{50/m/75/1(6)}$	924.75	1.7	**935**	4.5	1.10	0.00	23	3	5
$I_{50/m/75/2(7)}$	851.25	1.1	**857**	2.1	0.67	0.00	18	2	4
$I_{50/m/75/3(7)}$	863.00	1.3	**867**	2.8	0.46	0.00	20	2	4
$I_{50/m/75/4(7)}$	869.00	0.9	**879**	2.0	1.14	0.00	19	1	4
$I_{50/m/75/5(7)}$	971.50	1.0	**995**	7.7	2.36	0.00	22	3	4

Table 9 summarizes computational results of MT-BP for the four series of instances $S_{n/t}$ with $n \in \{50, 100\}$ and $t \in \{s, m\}$ (15 instances per set). The reported statistics are: the number of feasible solutions ("#Feas."), the number of optimal solutions ("#Opt."), the average number of trips per vehicle ("#trips/veh."), the average number of requests per trip ("#req/trip"), and the average number of requests per vehicle ("#req/veh."). Note that "#trips/veh.", "#req/trip" and "#req/veh." are computed only if a feasible solution is obtained.

Table 9 highlights the efficiency of MT-BP both on 50-customers instances and 100-customers instances. Note that theoretically, the maximum number of requests served in a trip can reach twice

Table 8: Results of the MT-BP algorithm on 100-customers instances

	RLB		UB		Gaps (%)		Trips		
Instance	RLB	Time^{RLB}	UB	Time	root gap	final gap	nb trips	min	max
$I_{100/s/25/1(15)}$	1868.24	48.1	-	-	-	-	-	-	-
$I_{100/s/25/2(14)}$	1746.50	86.2	-	-	-	-	-	-	-
$I_{100/s/25/3(15)}$	1712.00	75.5	**1718**	289.2	0.35	0.00	41	1	4
$I_{100/s/25/4(15)}$	1584.67	68.7	**1591**	142.9	0.40	0.00	39	1	5
$I_{100/s/25/5(15)}$	1753.50	47.3	**1757**	95.1	0.20	0.00	38	2	4
$I_{100/s/50/1(15)}$	1413.75	26.3	-	-	-	-	-	-	-
$I_{100/s/50/2(17)}$	1575.22	24.8	-	-	-	-	-	-	-
$I_{100/s/50/3(16)}$	1726.90	30.6	1751	-	-	0.70	35	1	3
$I_{100/s/50/4(14)}$	1588.67	36.5	-	-	-	-	-	-	-
$I_{100/s/50/5(14)}$	1537.50	29.4	**1547**	205.2	0.61	0.00	34	1	3
$I_{100/s/75/1(14)}$	1688.83	22.6	**1693**	507.5	0.25	0.00	40	2	4
$I_{100/s/75/2(14)}$	1595.88	33.8	**1606**	898.4	0.63	0.00	40	1	4
$I_{100/s/75/3(14)}$	1637.50	21.3	-	-	-	-	-	-	-
$I_{100/s/75/4(15)}$	1755.00	14.9	**1766**	4780.3	0.62	0.00	42	1	6
$I_{100/s/75/5(15)}$	1762.75	22.2	-	-	-	-	-	-	-
$I_{100/m/25/1(15)}$	1646.29	50.2	**1658**	1104.0	0.71	0.00	34	1	3
$I_{100/m/25/2(14)}$	1582.06	72.0	-	-	-	-	-	-	-
$I_{100/m/25/3(14)}$	1569.31	77.4	**1582**	1526.5	0.80	0.00	35	1	3
$I_{100/m/25/4(14)}$	1486.92	79.2	**1497**	971.6	0.67	0.00	34	1	4
$I_{100/m/25/5(15)}$	1599.61	50.9	**1612**	1077.3	0.77	0.00	34	1	4
$I_{100/m/50/1(14)}$	1307.82	33.9	1327	-	-	0.59	28	1	3
$I_{100/m/50/2(16)}$	1451.83	32.3	1472	-	-	0.20	28	1	3
$I_{100/m/50/3(16)}$	1605.81	30.1	**1614**	117.5	0.51	0.00	30	1	3
$I_{100/m/50/4(14)}$	1495.67	35.9	**1505**	117.1	0.62	0.00	29	1	3
$I_{100/m/50/5(14)}$	1481.67	32.3	**1484**	48.4	0.16	0.00	31	1	4
$I_{100/m/75/1(13)}$	1494.17	22.2	**1506**	123.3	0.79	0.00	32	1	4
$I_{100/m/75/2(14)}$	1470.99	27.7	**1485**	377.3	0.94	0.00	35	1	4
$I_{100/m/75/3(14)}$	1490.00	26.2	**1500**	786.3	0.67	0.00	30	1	4
$I_{100/m/75/4(14)}$	1549.28	16.1	**1561**	264.8	0.75	0.00	32	1	3
$I_{100/m/75/5(13)}$	1490.27	21.5	-	-	-	-	-	-	-

the vehicle capacity Q (Q outbound requests and Q inbound requests). For sedans ($Q = 2$), we can notice that the average filling rate of vehicles is not far from the maximal theoretical capacity. However, this average filling rate is relatively low for minivans ($Q = 4$). This behaviour can be explained by the value of the time horizon (4 hours) that makes the number of requests per hour relatively small. This is also due to the ride time that limits the detours to pickup or drop off of other patients. We can also notice that the number of trips per vehicle varies between 2 and 3. Note finally that the average number of requests per vehicle #req/veh is equal to #trips/veh $\times$ #req/trip.

Table 9: Summary of obtained solutions with MT-BP

Instance set	#feas.	#opt.	#trips/veh.	#req/trip	#req/veh.
$S_{50/s}$	15	15	2.95	2.29	6.76
$S_{50/m}$	15	15	2.74	2.55	7.00
$S_{100/s}$	8	7	2.62	2.59	6.78
$S_{100/m}$	13	11	2.20	3.16	6.95

6.5. Impact of the fleet size

In this section, we study the impact of the fleet size on the total cost and on the computing time. One objective is also to evaluate the opportunity to address this problem by first considering an infinite fleet of vehicles and second assigning the obtained trips to vehicles. This approach should simplify the solution process. Indeed, when the fleet is infinite, the set of feasible routes Ω can equivalently be reduced to the set of mono-trip routes: a multi-trip route can be considered as several mono-trip routes performed by several vehicles. To deal with the case of an infinite fleet, we thus implemented a mono-trip variant of MT-BP (following what is already explained in Section 6.1).

In order to perform this study, we selected four instances, one from each series of instances and run MT-BP with different fleet sizes. The selected instances are $I_{50/s/25/3(7)}$, $I_{50/m/25/4(7)}$, $I_{100/s/25/4(15)}$ and, $I_{100/m/25/5(15)}$. Note that for each instance and each fleet size MT-BP is able to prove either optimality or infeasibility. Obtained results are summarized in Tables 10, 11, 12 and, 13.

Tables 10, 11, 12 and, 13 contain one row per fleet size. Columns show for each fleet size: the solution cost ("cost"), the running time in seconds ("Time"), the total number of trips in the solution ("#trips"), the average number of trips per vehicle ("#trips/veh."), the average number of requests per trip ("#req/trip"), and the average number of requests per vehicle ("#req/veh."). The tables report the results with an unlimited number of vehicles and a single trip per vehicle (denoted ∞), then all fleet sizes $K \in \{K_1, \ldots, K_2\}$ where K_1 is the minimum fleet size required to obtain the minimal cost equivalent to an infinite size fleet, and K_2 is the maximum fleet size which enforces unfeasibility.

Table 10 shows that the best cost is 995 and is achievable with 10 vehicles. Then, reducing the fleet size first slightly increases the cost (from 995 to 1002 between 10 and 8 vehicles), then highly increases to 1054 with 7 vehicles. The instance is infeasible for $K \leq 6$. We notice that reducing the fleet size also impacts the computing time. An interesting remark is that the number of trips is not drastically impacted by the fleet size.

Results on tables 11, 12 and, 13 are similar. They show a slight impact on the cost when the fleet size is near K_1 and a high impact when the fleet size is close to K_2, especially for instance $I_{100/m/25/5}$ where the cost increases from 1631 to 1728 when passing from 13 to 12 vehicles.

Table 10: Impact of fleet size for instance $I_{50/s/25/3}$

Fleet size (K)	cost	Time	#trips	#trips/veh.	#req/trip	#req/veh.
∞	995	0.1	22	1.00	2.27	2.27
10	995	1.7	22	2.20	2.27	5.00
9	996	2.1	22	2.44	2.27	5.56
8	1002	4.2	22	2.75	2.27	6.25
7	1054	30.7	23	3.29	2.17	7.14
6	$\varnothing$	4.3	-	-	-	-

Table 11: Impact of fleet size for instance $I_{50/m/25/4}$

Fleet size (K)	cost	Time	#trips	#trips/veh.	#req/trip	#req/veh.
∞	920	0.4	22	1.00	2.27	2.27
10	920	4.2	22	2.20	2.27	5.00
9	922	9.6	22	2.44	2.27	5.56
8	932	66.9	22	2.75	2.27	6.25
7	1014	7	23	3.29	2.17	7.14
6	$\varnothing$	4.2	-	-	-	-

Table 12: Impact of fleet size for instance $I_{100/s/25/4}$

Fleet size (K)	cost	Time	#trips	#trips/veh.	#req/trip	#req/veh.
∞	1591	7.7	39	1.00	2.56	2.56
15	1591	142.7	39	2.60	2.56	6.67
14	1595	218.7	39	2.79	2.56	7.14
13	1607	418.1	39	3.00	2.56	7.69
12	1642	473.4	42	3.25	2.38	8.33
11	$\varnothing$	151.5	-	-	-	-

Table 13: Impact of fleet size for instance $I_{100/m/25/5}$

Fleet size (K)	cost	Time	#trips	#trips/veh.	#req/trip	#req/veh.
∞	1606	16.8	33	1.00	3.03	3.03
17	1606	156.8	33	1.94	3.03	5.88
16	1607	238.1	33	2.06	3.03	6.25
15	1612	1075.5	34	2.27	2.94	6.67
14	1617	844.2	34	2.43	2.94	7.14
13	1631	401.8	34	2.62	2.94	7.69
12	1728	1779.1	38	3.17	2.63	8.33
11	$\varnothing$	82.0	-	-	-	-

As expected, the computing time is significantly lower when the fleet is infinite and that the mono-trip variant of MT-BP is used. This approach, combined with a post-assignment of trips to vehicles, is clearly to be preferred when the fleet is large enough. However, the tables show its limits. Indeed, it would never give a solution with less than K_1 vehicles. Our results suggest that reducing the fleet size by a few vehicles by allowing multiple trips does not have a high impact on transportation costs. Furthermore, if the number of available vehicles is a critical resource for the company providing the transportation service, the use of multiple trips can be necessary to satisfy all requests. It is also important to note that K_1 and K_2 are not known in advance and depend on the instance characteristics. Hence, it is difficult to be sure in advance that the solution provided by the mono-trip variant would comply with the fleet size.

6.6. Other sensitivity analyses

In this section, we present other sensitivity analyses. We generated new challenging instances for the MTMPD-RD based on instances presented in Section 6.2 with modified parameters to assess the limits of the MT-BP algorithm.

Let us recall that, for all instances presented in Section 6.2, the time horizon is $[0\,min., 240\,min.]$ which corresponds to a working period of 4 hours, and a ride is considered acceptable for a customer if its duration is the time needed for a direct ride between the pickup and drop-off locations plus 20 minutes (*i.e.*, 20 minutes corresponds to the maximal acceptable detour).

In this section we consider a larger time horizon and a larger acceptable ride time to evaluate how the MT-BP algorithm scales up. We propose three new sets of instances:

- Larger acceptable ride time: allowing detours of 40 minutes and keeping a time horizon of 240 minutes.
- Larger time horizon: $[0\,min., 480\,min.]$ (8 hours) and keeping acceptable detours of 20 minutes.
- Both a larger time horizon $[0\,min., 480\,min.]$ and larger acceptable detours (40 minutes).

Let E_0 be the initial set of instances presented in Section 6.2. E_0 is composed of 60 instances: 30 instances with 50 requests and 30 instances with 100 requests. The three new sets are derived from E_0, especially customers remain the same, and are defined as follows:

- E_1 : instances of E_0 with modified release and due dates to extend the maximal detour from 20 to 40 minutes: $r_i \leftarrow r_i - 10$ and $d_i \leftarrow d_i + 10$; however, if the new values are not consistent with the conditions defined in Section 6.2, we rather set $r_i \leftarrow r_i$ and $d_i \leftarrow d_i + 20$, or $r_i \leftarrow r_i - 20$ and $d_i \leftarrow d_i$, when the new release date is too small or the new due date too large, respectively.
- E_2 : instances of E_0 with the time horizon enlarged from $[0\,min., 240\,min.]$ to $[0\,min., 480\,min.]$ and the release and due dates regenerated.
- E_3 : instances of E_2 with release and due dates modified similarly to instance set E_1.

Fleet sizes are determined the same way as they were for instances of set E_0. We run the MT-BP algorithm on all these new instances. Similarly to Table 9 for instances of set E_0, tables 14, 15 and 16 summarize computational results for instances of sets E_1, E_2 and, E_3.

Tables 14 and 16 show the limits of MT-BP to solve instances with large maximal detour allowed. Both for E_1 and E_3, optimality could not be proven for three 50-customers instances. For 100-customers instances, MT-BP found feasible solutions for 5 and 4 for instances respectively in

Table 14: Summary of obtained solutions with MT-BP for the E_1 instances

Instance set	#feas.	#opt.	#trips/veh.	#req/trip	#req/veh.
$S_{50/s}$	15	13	3.16	2.58	8.15
$S_{50/m}$	15	14	2.64	3.64	9.62
$S_{100/s}$	0	0	-	-	-
$S_{100/m}$	5	2	2.13	4.50	9.62

Table 15: Summary of obtained solutions with MT-BP for the E_2 instances

Instance set	#feas.	#opt.	#trips/veh.	#req/trip	#req/veh.
$S_{50/s}$	15	15	5.29	2.15	11.36
$S_{50/m}$	15	15	4.69	2.25	10.56
$S_{100/s}$	10	5	4.44	2.53	11.24
$S_{100/m}$	13	10	3.77	2.80	10.57

Table 16: Summary of obtained solutions with MT-BP for the E_3 instances

Instance set	#feas.	#opt.	#trips/veh.	#req/trip	#req/veh.
$S_{50/s}$	15	14	5.45	2.50	13.64
$S_{50/m}$	15	13	3.96	3.32	13.16
$S_{100/s}$	2	1	4.77	3.23	15.38
$S_{100/m}$	2	1	3.19	3.92	12.50

E_1 and E_3, and was able to prove optimality for 2 instances in both sets. Similarly to the issue of large time windows for the VRPSPDTW, enlarging the maximal detour in instances of the MTMPD-RD increases the number of feasible solutions in the dynamic programming process when solving the subproblems and drastically impacts the effectiveness of the solution method. Table 15 shows results for E_2 comparable to those of E_0 (see Table 9) with all 50-customers instances solved to optimality and 23 feasible solutions found (of which 15 are proven optimal) for 100-customers instances. These results highlight that the time horizon doesn't seem to have an important impact on efficiency of the MT-BP algorithm. However, increasing the maximal detour highly deteriorates its efficiency.

7. Conclusion

In this paper we introduced a new problem arising from the context of non-emergency transportation of patients. The main originality of the studied problem is that there are two types of requests, a patient is either picked-up at home and transported to the hospital or the other way around. Requests are thus always connected to the hospital, and routes combine both types of requests. In addition, the quality of service is preserved with riding time constraints that enforce the vehicles to regularly come back to the hospital. Consequently, the problem can be addressed either as a special Pickup and Delivery Problem or as a special multi-trip VRP. We call it the Multi-Trip Vehicle Routing Problem with Mixed Pickup and Delivery, and Release and Due dates (MTMPD-RD).

The main goal of the paper was then to evaluate if tackling the problem as a multi-trip VRP should be preferred. We proposed a branch-and-price algorithm (MT-BP). The complex time and capacity constraints of the problem make the definition and the management of resources complex in the pricing problem. Also, they implied some difficulties in the branching scheme.

As the MTMPD-RD is a new problem, we generated a new set of realistic instances. We also implemented a branch-and-price algorithm based on the pickup and delivery formulation (PD-BP). Experiments showed that the MT-BP outperforms the PD-BP and prove the efficiency of a multi-trip model for this problem. MT-BP proved its efficiency by solving to optimally instances involving up to 100 requests.

To further validate the MT-BP, we adapted it for the solution of the VRPSPDTW. Our experiments on benchmark instances closed the benchmark for small instances and closed four 100-customers instances. We also detected that an infeasible solution was published in the literature. In addition, we evaluated the opportunity to tackle the problem as a mono-trip vehicle routing problem followed by a trip-to-vehicle assignment. Experiments show the computational interest but also the limits of this standpoint. Finally, we conducted other sensitivity analyses generating new challenging instances with a larger time horizon and/or a larger acceptable ride time. Results show that the computational time of the MT-BP algorithm is strongly impacted when the maximal detour is increased, but remains relatively stable for larger time horizons.

A first perspective for this work would be to improve the solution method, by adding valid cuts or using a bidirectional approach for the pricing problem. Also, one might propose heuristics to solve the MTMPD-RD to be able to solve larger instances in a short amount of time, which would be appreciated in real-life applications. An important real-life constraint is also missing in this study. Results show a strong imbalance in the duration of routes. It is inappropriate in our context because the transportation is subcontracted to independent paramedics that should be considered equitably. In addition, the easiness of patient management may vary depending on their pathology or their lodging accessibility, which also potentially generates inequity among drivers. An interesting perspective would thus be to integrate equity into the problem with both regards.

Acknowledgments

This work is part of project FITS - "Flexible and Intelligent Transportation Systems", supported by the French National Research Agency (ANR - Agence Nationale de la Recherche) under grant ANR-18-CE22-0014.

References

Angelelli, E., Mansini, R., 2002. The vehicle routing problem with time windows and simultaneous pick-up and delivery, in: Quantitative approaches to distribution logistics and supply chain management. Springer, pp. 249–267.

Battarra, M., Cordeau, J.F., Iori, M., 2014. Chapter 6: pickup-and-delivery problems for goods transportation, in: Vehicle Routing: Problems, Methods, and Applications, Second Edition. SIAM, pp. 161–191.

Berbeglia, G., Cordeau, J.F., Gribkovskaia, I., Laporte, G., 2007. Static pickup and delivery problems: a classification scheme and survey. Top 15, 1–31.

Cattaruzza, D., Absi, N., Feillet, D., 2016. Vehicle routing problems with multiple trips. 4OR 14, 223–259.

Cordeau, J.F., 2006. A branch-and-cut algorithm for the dial-a-ride problem. Operations Research 54, 573–586.

Cordeau, J.F., Laporte, G., 2003. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transportation Research Part B: Methodological 37, 579–594.

Dantzig, G.B., Ramser, J.H., 1959. The truck dispatching problem. Management science 6, 80–91.

Feillet, D., Dejax, P., Gendreau, M., Gueguen, C., 2004. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. Networks: An International Journal 44, 216–229.

Fleischmann, B., 1990. The vehicle routing problem with multiple use of vehicles. Forschungsbericht Fachbereich Wirtschaftswissenschaften, Universität Hamburg .

Fogue, M., Sanguesa, J.A., Naranjo, F., Gallardo, J., Garrido, P., Martinez, F.J., 2016. Non-emergency patient transport services planning through genetic algorithms. Expert Systems with Applications 61, 262–271.

Garaix, T., Artigues, C., Feillet, D., Josselin, D., 2010. Vehicle routing problems with alternative paths: An application to on-demand transportation. European Journal of Operational Research 204, 62–75.

Hernandez, F., Feillet, D., Giroudeau, R., Naud, O., 2014. A new exact algorithm to solve the multi-trip vehicle routing problem with time windows and limited duration. 4or 12, 235–259.

Hernandez, F., Feillet, D., Giroudeau, R., Naud, O., 2016. Branch-and-price algorithms for the solution of the multi-trip vehicle routing problem with time windows. European Journal of Operational Research 249, 551–559.

Hof, J., Schneider, M., 2019. An adaptive large neighborhood search with path relinking for a class of vehicle-routing problems with simultaneous pickup and delivery. Networks 74, 207–250.

Koc, C., Karaoglan, I., 2011. A branch and cut algorithm for the vehicle routing problem with multiple use of vehicles, in: 41st International Conference on Computers & Industrial Engineering, pp. 554–559.

Koç, Ç., Laporte, G., Tükenmez, İ., 2020. A review of vehicle routing with simultaneous pickup and delivery. Computers & Operations Research 122, 104987.

Lim, A., Zhang, Z., Qin, H., 2017. Pickup and delivery service with manpower planning in hong kong public hospitals. Transportation Science 51, 688–705.

Liu, M., Luo, Z., Lim, A., 2015. A branch-and-cut algorithm for a realistic dial-a-ride problem. Transportation Research Part B: Methodological 81, 267–288.

Luo, Z., Liu, M., Lim, A., 2019. A two-phase branch-and-price-and-cut for a dial-a-ride problem in patient transportation. Transportation Science 53, 113–130.

Mingozzi, A., Roberti, R., Toth, P., 2013. An exact algorithm for the multitrip vehicle routing problem. INFORMS Journal on Computing 25, 193–207.

Molenbruch, Y., Braekers, K., Caris, A., 2017. Typology and literature review for dial-a-ride problems. Annals of Operations Research 259, 295–325.

Mosheiov, G., 1998. Vehicle routing with pick-up and delivery: tour-partitioning heuristics. Computers & Industrial Engineering 34, 669–684.

Paradiso, R., Roberti, R., Laganá, D., Dullaert, W., 2020. An exact solution framework for multitrip vehicle-routing problems with time windows. Operations Research 68, 180–198.

Parragh, S.N., Cordeau, J.F., Doerner, K.F., Hartl, R.F., 2012. Models and algorithms for the heterogeneous dial-a-ride problem with driver-related constraints. OR spectrum 34, 593–633.

Privé, J., Renaud, J., Boctor, F., Laporte, G., 2006. Solving a vehicle-routing problem arising in soft-drink distribution. Journal of the Operational Research Society 57, 1045–1052.

Qu, Y., Bard, J.F., 2013. The heterogeneous pickup and delivery problem with configurable vehicle capacity. Transportation Research Part C: Emerging Technologies 32, 1–20.

Qu, Y., Bard, J.F., 2015. A branch-and-price-and-cut algorithm for heterogeneous pickup and delivery problems with configurable vehicle capacity. Transportation Science 49, 254–270.

Shi, Y., Boudouh, T., Grunder, O., 2018. An efficient tabu search based procedure for simultaneous delivery and pick-up problem with time window. IFAC-PapersOnLine 51, 241–246.

Shi, Y., Zhou, Y., Boudouh, T., Grunder, O., 2020. A lexicographic-based two-stage algorithm for vehicle routing problem with simultaneous pickup–delivery and time window. Engineering Applications of Artificial Intelligence 95, 103901.

Solomon, M.M., 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations research 35, 254–265.

Subramanian, A., Uchoa, E., Pessoa, A.A., Ochi, L.S., 2013. Branch-cut-and-price for the vehicle routing problem with simultaneous pickup and delivery. Optimization Letters 7, 1569–1581.

Toth, P., Vigo, D., 2014. Vehicle routing: problems, methods, and applications. SIAM.

Wang, C., Mu, D., Zhao, F., Sutherland, J.W., 2015. A parallel simulated annealing method for the vehicle routing problem with simultaneous pickup–delivery and time windows. Computers & Industrial Engineering 83, 111–122.

Wang, H.F., Chen, Y.Y., 2012. A genetic algorithm for the simultaneous delivery and pickup problems with time window. Computers & industrial engineering 62, 84–95.

Zhang, Z., Liu, M., Lim, A., 2015. A memetic algorithm for the patient transportation problem. Omega 54, 60–71.