European Journal of Operational Research 280 (2020) 953–969

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Production, Manufacturing, Transportation and Logistics

Recovery management for a dial-a-ride system with real-time disruptions

Célia Paquay[a,b,*], Yves Crama[a], Thierry Pironet[a]

[a] HEC Liège, Management School of the University of Liège, Liège, Belgium
[b] School of Business and Economics, Maastricht University, Maastricht, the Netherlands

ARTICLE INFO

Article history:
Received 31 May 2019
Accepted 5 August 2019
Available online 9 August 2019

Keywords:
Transportation
Dial-a-ride
Reactive algorithm
Real-time disruptions
Health care

ABSTRACT

The problem considered in this work stems from a non-profit organization in charge of door-to-door passenger transportation for medical appointments. Patients are picked up at home by a driver and are then dropped at their appointment location. They may also be driven back home at the end of their appointment. Some patients have specific requirements, e.g., they may require an accompanying person or a wheelchair. Planning such activities gives rise to a so-called dial-a-ride problem. In the present work, it is assumed that the requests assigned to the drivers have been selected, and the transportation plan has been established for the next day. However, in practice, appointment durations may vary due to unforeseen circumstances, and some transportation requests may be modified, delayed or canceled during the day. The aim of this work is to propose a reactive algorithm which can adapt the initial plan in order to manage the disruptions and to take care of as many patients as possible in real-time. The plan should be modified quickly when a perturbation is observed, without resorting to major changes which may confuse the drivers and the patients. Several recourse procedures are defined for this purpose. They allow the dispatcher to temporarily delete a request, to insert a previously deleted request, or to permanently cancel a request. Simulation techniques are used to test the approach on randomly generated scenarios. Several key performance indicators are introduced in order to measure the impact of the disruptions and the quality of the solutions.

1. Introduction

The problem considered in this article stems from a non-profit organization in charge of door-to-door transportation of patients for medical appointments. Patients are picked up at home and are then dropped at their appointment location (outbound requests). Some patients must also be picked up at the end of their appointment to be driven back home (inbound requests). Each appointment has a predefined start time and end time, but in practice, its duration may vary (most often, increase) unpredictably, and this information becomes available for the dispatching office when the patient actually comes out.

This type of situation gives rise in the scientific literature to a so-called *dial-a-ride problem* (DARP). The classic dial-a-ride problem mainly arises in door-to-door transportation services for elderly or disabled people (Cordeau & Laporte (2007)). The DARP consists in planning a collection of journeys, i.e., of routes and associated schedules, for a fleet of vehicles so as to satisfy the patient outbound and inbound requests while meeting various constraints, such as: time window constraints (the pickups and the deliveries must be performed within given time intervals), maximum ride time for patients and drivers, and capacity constraints of vehicles. The objective function of the DARP can differ from one application to the next, and may include economic or service level considerations. For instance, it might be to maximize the number of satisfied requests, to minimize the number of required vehicles, or to minimize the waiting time of the patients (Paquette, Cordeau, & Laporte, 2009). More details about the DARP features are provided in Section 2.

In the application targeted in this work, the primary aim of the organization is to maximize the number of requests that it can handle with its own fleet. Accordingly, every evening, the dispatching office selects the transportation requests that will be served on the next day and it establishes the route and corresponding schedule of each vehicle (Cappart, Thomas, Schaus, & Rousseau, 2018; Liu, Aleman, & Beck, 2018). However, some events may occur and lead to unexpected changes when the plan is actually implemented. For instance, the real duration of an appointment may differ from the forecast; the physician may cancel the appointment;

* Corresponding author at: HEC Liège, Management School of the University of Liège, Liège, Belgium.
E-mail addresses: cpaquay@uliege.be (C. Paquay), yves.crama@uliege.be (Y. Crama), thierry.pironet@uliege.be (T. Pironet).

https://doi.org/10.1016/j.ejor.2019.08.006
0377-2217/

the patient may decide to go home by her own means; and so forth. In practice, along the day, the dispatcher is informed of each disruption either by a phone call announcing an appointment cancellation or because a patient shows up later than initially planned. The dispatcher must make a decision as to how to adapt the plan in response to this incoming information. The recovery decision can be especially difficult due the fact that the schedules produced by the initial optimization phase are usually tight, since the organization tends to maximize the number of requests to be served. Our work aims at adapting the current plan in real time so as to manage the disruptions while accounting for three potentially conflicting objectives, namely: satisfying all patient requests, to the best possible extent; limiting the planning changes, so as to avoid confusion for the drivers and patients; and minimizing the patient excess journey duration, as a measure of service quality. Moreover, the dispatcher should be able to react quickly, so as to give instructions to the drivers and to take care of the patients as soon as possible, and to define the recourse action before the next disruption occurs.

The article is organized as follows. Section 2 positions our contribution with respect to the DARP literature. In Section 3, the statement of the problem and the representation of a solution are discussed in detail. Section 4 describes the nature of the real-time disruptions and the associated recourse actions. Section 5 presents the simulation settings. The results of the computational experiments are discussed in Section 6. Conclusions are finally drawn in Section 7.

2. Literature review and positioning of the contribution

Dial-a-ride problems are motivated by real-life applications. This results in a broad variety in the definition of constraints and of objective functions, as well as in the assumptions about the nature of available information. These features are described in detail in recent surveys by Molenbruch, Braekers, and Caris (2017) or Ho et al. (2018). We only review them here very briefly, with the goal to position our work with respect to earlier contributions.

2.1. Constraints

All versions of the DARP share common types of constraints: pickup and delivery operations have time windows (usually, short ones) in which the requests should be handled (e.g., Coslovich, Pesenti, & Ukovich, 2006; Jaw, Odoni, Psaraftis, & Wilson, 1986), all vehicles have a limited capacity (identical or different for each vehicle if the fleet is heterogeneous) and sometimes several types of seats, each patient may have a maximum ride time, and the drivers may have a maximum allowed working time. The ride time may be limited to an identical predetermined value for all the patients, or to a value which is proportional to the direct ride time (a detour may be admissible for a long trip, but not for a short one) (Molenbruch et al., 2017). Our problem statement features most of the above constraints. More details will be provided in subsequent sections (in particular, Section 3 and Section 5.2).

2.2. Objective functions

As explained in Molenbruch et al. (2017) and Ho et al. (2018), the objective function of the problem varies with the application. According to Paquette et al. (2009), the service quality in the DARP can be defined according to company specifications (technical approach), leading to *operational objectives*, or according to customer perceptions (customer-based approach), leading to *quality-related objectives*. Examples of both types of objectives are listed in Table 1.

In the DARP, both types of objectives must be taken into consideration, since vehicles and patients are concerned. However, these different objectives can be conflicting with each other: for instance, decreasing the patient waiting time may tend to increase the number of vehicles needed, or maximizing the occupancy rate of the vehicles may tend to deteriorate the patients' ride time.

When combining several objectives, three strategies are observed in the literature (Ho et al., 2018; Molenbruch et al., 2017). The first one is to consider a weighted sum of the objectives. The difficulty is then to set the weights of the different elements of the sum. The second strategy is to define a hierarchical (or lexicographic) objective function. This strategy might be relevant when one objective has a higher priority than the other ones. The third approach is to obtain the Pareto frontier of the problem. The decision maker should then choose a preferred solution among the (sometimes huge) set of non-dominated solutions. Because of its complexity, this approach may not be suitable for problems requiring calculations in real-time (Ho et al., 2018).

In this paper, the problem we face is to generate recourse actions leading to new plans in order to manage real-time disruptions. We do not compute the initial plan, which we assume to be given to us (it is currently generated by a self-standing application), but we manage a reactive plan as unexpected events unfold over time. Our main and foremost goal is to maintain a feasible plan while considering the current system status and accounting for new revealed information. Thus, we mostly strive to maximize the number of served requests (or "throughput", Pillac, Gendreau, Guéret, & Medaglia, 2013). We favor, albeit in a more informal way, plans that do not differ too much from the current one. This is achieved by relying on local search operators to recover from disruptions. As a secondary objective reflecting the quality of the modified plan, when feasibility is restored, we try to minimize the excess journey time ratio of the patients, i.e., the ratio between the actual journey duration and the shortest possible journey duration. However, as explained in Pillac et al. (2013) and already mentioned in Section 1, making decisions in real time often enforces trade-offs between speed and quality of the responses.

2.3. Nature of the available information

Similarly to Pillac et al. (2013) and Psaraftis, Wen, and Kontovas (2016), a taxonomy for DARPs based on the nature of the available information is proposed in Ho et al. (2018) :

1. the decisions are made a priori and cannot be changed later (static), or the decisions can be modified due to new revealed information (dynamic);
2. the information, when received, is known with certainty (deterministic) or is unknown or uncertain at the time when decisions are made (stochastic).

This taxonomy leads to four basic DARP categories represented in Table 2. In this table is also indicated the number of articles studying each category among 86 published mainly since 2007 and surveyed by Ho et al. (2018).

In our targeted application, information is available at the beginning of the day, but some disruptions (delays, cancellations) are observed in real-time and require changes in the plan. Thus, the DARP considered in this work is dynamic and deterministic. As explained in Berbeglia, Cordeau, and Laporte (2010), two approaches can be distinguished to deal with the dynamic case: (1) solving the static problem each time a disruption occurs, meaning a complete reoptimization of the transportation plan, or (2) applying the static algorithm only once to obtain the initial plan, then using fast heuristic techniques to update the plan each time a disruption occurs. (Note that the objectives of the algorithms used to create the initial plan and those of the reactive heuristics may be different.)

Table 1
Examples of objective functions in DARP.

Operational	Quality-related
Minimize fleet size	Minimize users' ride time
Maximize vehicle occupancy rate	Minimize users' excess ride time
Minimize vehicle idle time	Minimize users' waiting time
Minimize the time needed to complete all requests	Minimize users' total journey time (i.e., waiting and riding time)
Maximize the total profit associated with selected requests	Minimize number of requests served while a user is on board
Minimize taxi costs to cover fleet shortages	Maximize number of requests served
Minimize vehicle emissions	
Minimize total vehicle distance	
Minimize total driver wage	

Table 2
Taxonomy of dial-a-ride problems from Ho et al. (2018) and number of articles studying each category among 86 published mainly since 2007.

		Information is known with certainty?	
		Yes	No
Decisions can be modified in response to new information revealed after time 0?	No	Static and deterministic *64*	Static and stochastic *3*
	Yes	Dynamic and deterministic *10*	Dynamic and stochastic *9*

In practice, the first approach can be time consuming and tends to deeply modify the current plan, which can confuse or disturb the drivers and the patients. For these reasons, we have adopted the second approach. The present work mainly focuses on a reactive process which can be used to quickly adapt the initial plan in response to real-time disruptions.

2.4. Positioning and contribution of the paper

Even though the constraints and the objective function considered in the present work are classic for a DARP, the contribution of this work is threefold. First, we show how to effectively and efficiently handle the information dynamics of the application. The development of disruption management methods is one of the promising areas for future research highlighted in Ho et al. (2018). The decision maker, in this case the dispatching office, needs to be able to quickly identify a new plan, in particular before the next disruption occurs. This explains why we focus on heuristic methods. A second reason for the adoption of heuristics is that the changes to be brought to the current plan have to be as small as possible in order to avoid unpleasant disturbances for the drivers and the patients. (This is similar in spirit to the concept of "limited recovery possibilities" introduced by Liebchen, Lübbecke, Möhring, & Stiller, 2009.) We achieve this goal by exploring small neighborhoods of the current solution and very progressively enlarging these neighborhoods in order to restore feasibility when a perturbation occurs. We will show in the experiments that the use of simple operators is often sufficient to handle the disruptions, without considerably affecting the quality of the plan. Finally, a third advantage of the heuristic proposed here is its flexibility. Each DARP has its own specific features, and thus being able to easily tailor the algorithm in order to account for these features represents a considerable benefit.

Our second main contribution is related to the different types of disruptions taken into consideration for the initial requests. As explained in Ho et al. (2018), most of the articles dealing with dynamic and deterministic DARPs only focus on inserting new user requests. In our study, the variation of the appointment duration for some patients may cause conflicting situations. In particular, a large delay may lead the driver to depart from the pickup point without the patient, so as to limit the consequences for the other patients. From an operational point of view, this decision results in a temporary removal of two subrequests (one pickup and one delivery for the delayed patient); but later in time (when the delayed patient finally completes the appointment), the reinsertion of these two subrequests in a route must be attempted. Removals and reinsertions of requests have been previously used as local search operators in the design of heuristic algorithms for DARP or VRP problems; see, e.g., Cordeau and Laporte (2003). In our context, however, the removals and reinsertions are not algorithmic ingredients, but are the results of disruptions. As such, they modify the set of requests and the time windows defining the instance itself.

Finally, the last contribution of our work lies in the different key performance indicators introduced to analyze the experiment results in Section 6. In particular, the frequency and the severity of the risks are analyzed when the probabilities of cancellations and the delays vary.

3. Problem description

In this section, we provide a formal description of the problem at hand and we discuss conditions for the existence of a feasible schedule.

3.1. Notations and solution representation

The aim of the procedures proposed in this paper is to be able to react quickly and effectively in case of real-time disruptions. As explained above, we assume that the requests to be served have been selected beforehand and that a feasible plan is provided as input. Hence, the input of the problem consists of a set of *vehicles*, a set of patient *requests*, and an initial *plan*. Here, a plan is a collection of *routes* and their associated *schedule* of activities (Mitrović-Minić, Krishnamurti, & Laporte, 2004). Each route is an ordered sequence of pickup and delivery locations to be visited by a vehicle. The schedule associated with a route is the list of arrival and departure times for each location in the route. In the sequel, the terms plan and solution will be used indifferently. We now detail the data and decisions of the problem. Each vehicle $j \in \mathcal{J} = \{1, ..., J\}$ is characterized by a number of seats, a number of places for wheelchairs, and the driver working hours $[\alpha_j, \beta_j]$.

Each request consists of two associated *subrequests*: a pickup (or loading) subrequest and a delivery (or unloading) subrequest. The route ρ_j associated with a vehicle j is described by a sequence $(l_{j,0}, r_{j,1}, \ldots, r_{j,k}, \ldots, r_{j,K_j}, l_{j,K_j+1})$, where $l_{j,0}$ is the location of the starting depot, $r_{j,1}, \ldots, r_{j,k}, \ldots, r_{j,K_j}$ are the subrequests

Table 3
Notations.

Notation	Definition
$[\alpha_j, \beta_j]$	Working period of the driver of vehicle j
$t_{l,l'}$	Travel time from location l to location l'
ρ_j	Route of vehicle j
r_k	Subrequest k
l_k	Location of subrequest k
s_k	Service duration of subrequest k
$[a_k, b_k]$	Time window for the start of the execution of subrequest k
A_k	Arrival time at l_k for the execution of r_k
D_k	Departure time at l_k for the execution of r_k
H_k	Start time of execution of r_k

served along the route, and l_{j,K_j+1} is the location of the final depot, which can be identical or not to $l_{j,0}$. (All these parameters depend on the route ρ_j, but when no confusion can arise, we will often omit the index j for the sake of simplicity.) The earliest departure time from the starting depot is equal to α_j, and the latest arrival time at the final depot is equal to β_j.

Each subrequest r_k has the following features: a location l_k, a time window $[a_k, b_k]$ during which the execution of the subrequest (loading or unloading the patient under consideration) must be started, and a service duration s_k which represents the time needed for loading or unloading the patient (s_k is also called dwell time in Jaw et al., 1986; Psaraftis, 1986; Savelsbergh & Sol, 1995). Additional parameters may specify whether the patient requires a wheelchair or an accompanying person. Two consecutive subrequests r_k, r_{k+1} can take place at different locations ($l_k \neq l_{k+1}$) or at the same location ($l_k = l_{k+1}$). Between the execution of two consecutive subrequests, there can be waiting time (if the locations are identical) or travel time (if the locations are different). The driving time from location l to location l' is denoted as $t_{l,l'}$. The subrequest time window $[a_k, b_k]$ depends on the appointment time of the patient and on the travel time between her pickup location and her delivery location. It essentially expresses that the patient should be on time for her appointment and should be brought back home within a maximum time limit after the end of her appointment. (For our particular case study, more details about the construction of the subrequest time windows are provided in Section 5.2 – see Table 5.)

Two associated subrequests (pickup and delivery) should obviously be assigned to the same vehicle, and the pickup should precede the corresponding delivery. For each subrequest r_k performed at location l_k, we define a schedule by three time instants: the arrival time A_k at l_k, the start time H_k of the execution of r_k (load or unload), and the departure time D_k from l_k. By extension, D_0 is the departure time from the starting depot location l_0 ($D_0 \geq \alpha_j$), and A_{K+1} is the arrival time at the final depot location l_{K+1} ($A_{K+1} \leq \beta_j$). Note that the time-related decision variables are represented by capital letters. Table 3 summarizes the notations.

The associated schedule of a route (say $l_0, r_1, r_2, \ldots, r_6, l_0$) can be graphically represented in a time-space coordinate system as illustrated in Fig. 1. The filled rectangles in this representation are associated with the subrequests performed in each route; the length of each rectangle corresponds to the subrequest service duration. Horizontal line segments mean that the vehicle remains motionless for a while (so, for instance, the first three subrequests take place at the same location $l_1 = l_2 = l_3$). Line segments with a non-zero slope indicate that the vehicle moves from one location to the next.

3.2. Feasible solutions

A solution is feasible if the routes and the schedules are feasible. The feasibility of the routes consists in satisfying the following constraints:

(RF1) the capacity of the vehicles must be respected at anytime in any route;
(RF2) the pickup and the delivery of a same request are assigned to the same vehicle;
(RF3) the pickup of a request must precede its delivery.

In terms of time, a schedule is feasible if:

(SF1) each vehicle j leaves its starting depot not earlier than time α_j and returns to its final depot not later than time β_j;
(SF2) the time windows for picking up and delivering each patient are respected ($H_k \in [a_k, b_k]$ for both subrequests);
(SF3) there is enough time for the vehicle to move from one location to the next ($D_k + t_{l_k,l_{k+1}} = A_{k+1}$);
(SF4) the execution of a subrequest starts after the arrival at the required location ($A_k \leq H_k$) and ends before the departure from the location ($D_k \geq H_k + s_k$).

3.3. Earliest and latest arrival and departure times

Consider a feasible route $\rho_j = (l_0, r_1, \ldots, r_K, l_{K+1})$. As explained by Savelsbergh (1992) and Mitrović-Minić and Laporte (2004), and as suggested by conditions (SF1)–(SF4), the scheduled arrival time A_k at location l_k must be in an interval $[\underline{A}_k, \overline{A}_k]$, where $\underline{A}_k$ is the *earliest arrival time* and $\overline{A}_k$ is the *latest arrival time* at location l_k. Similarly, the scheduled departure time D_k from location l_k must be between an *earliest departure time* $\underline{D}_k$ and a *latest departure time* $\overline{D}_k$.

A forward pass through the requests served along the route determines the earliest arrival and departure times at each location, while a backward pass determines the latest arrival and departure times (this is akin to critical path computations in project scheduling). The computations are initialized using the driver working hours $[\alpha_j, \beta_j]$: $\underline{D}_0 = \alpha_j$ and $\overline{A}_{K+1} = \beta_j$ (see condition (SF1)). Next, taking into account the time windows $[a_k, b_k]$, $k = 1, \ldots, K$, the recursive definitions of Mitrović-Minić and Laporte (2004) can be adapted as follows:

- Earliest arrival and departure time with a forward pass:

$$\begin{cases} \underline{A}_k = \underline{D}_{k-1} + t_{l_{k-1},l_k} & \forall k = 1, \ldots, K+1 \\ \underline{D}_k = \max\{a_k, \underline{A}_k\} + s_k & \forall k = 1, \ldots, K \end{cases}$$

(the earliest arrival time at the kth subrequest location, $\underline{A}_k$, is the earliest time when the kth location may be reached considering that the $k-1$ previous locations have been visited; the earliest departure time from the kth subrequest location, $\underline{D}_k$, is then derived from $\underline{A}_k$ and from the subrequest time window $[a_k, b_k]$).

- Latest arrival and departure time with a backward pass:

$$\begin{cases} \overline{D}_k = \overline{A}_{k+1} - t_{l_k,l_{k+1}} & \forall k = 0, \ldots, K \\ \overline{A}_k = \min\{\overline{D}_k - s_k, b_k\} & \forall k = 1, \ldots, K \end{cases}$$

(the latest departure time from the kth subrequest location, $\overline{D}_k$, is the latest possible time when the vehicle can leave the location to be on time for the following subrequests; the latest arrival time at the kth subrequest location, $\overline{A}_k$, is then derived from the latest departure time $\overline{D}_k$ and from the subrequest time window $[a_k, b_k]$).

Given a feasible route (in the sense of conditions (RF1)–(RF3)), necessary and sufficient conditions for the existence of a feasible schedule are: for all subrequests r_k,

$$[\underline{A}_k, \overline{A}_k] \neq \emptyset,\ [\underline{D}_k, \overline{D}_k] \neq \emptyset,\ [\max\{a_k, A_k\}, \min\{D_k - s_k, b_k\}] \neq \emptyset, \quad (1)$$

where the last condition implies that one can find a starting time H_k for the execution of subrequest r_k that lies in both intervals $[A_k, D_k - s_k]$ and $[a_k, b_k]$. By default, when (1) holds for all k,

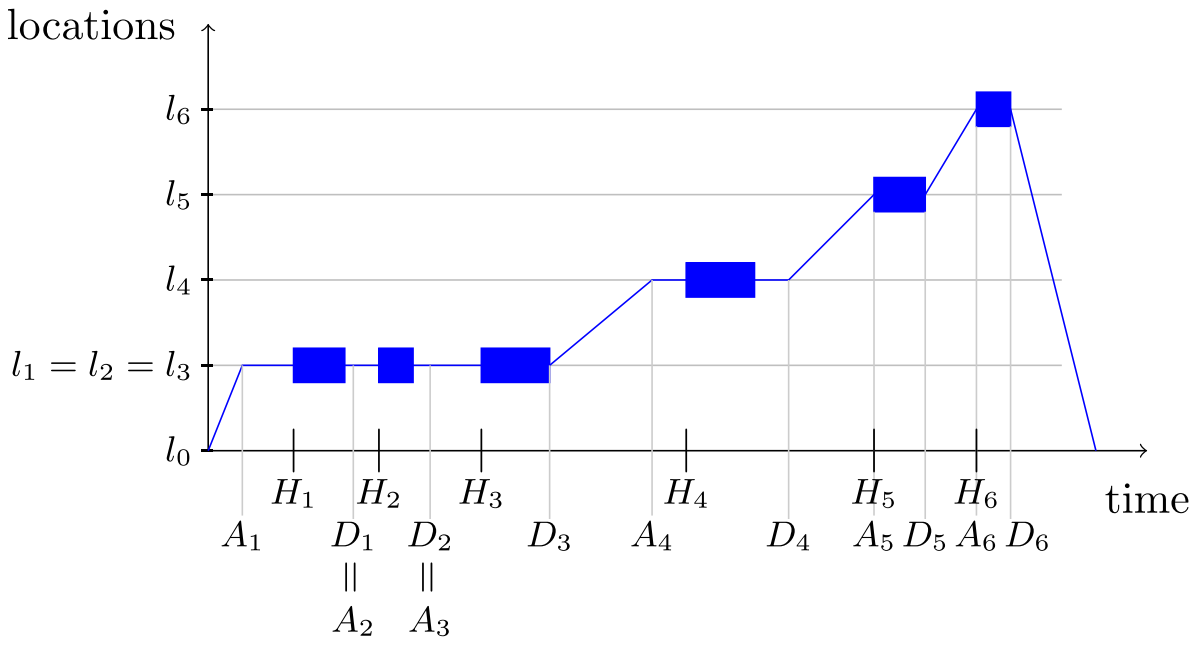


Fig. 1. Schedule representation and time-related variables.

we generate a feasible plan by considering that every action is performed as soon as possible, that is, $A_k = \underline{A}_k, D_k = \underline{D}_k$ and $H_k = \max\{a_k, A_k\}$. This strategy is very natural in practice: the driver does not keep a patient waiting more than necessary, and once the patient is on board, the driver can leave straight away. Intuitively, the strategy aims at minimizing the excess journey time of the patients.

Similarly to project scheduling, the *slack time* of subrequest r_k on a route can be defined as the difference between the latest departure time and the earliest arrival time at location l_k, i.e., the maximum amount of time that a driver can spend at the location:

$$slack(k) = \overline{D}_k - \underline{A}_k. \tag{2}$$

The larger the slack time, the easier it is to find a feasible solution with respect to time constraints. For this reason, the average slack of a solution will be used as selection criterion in Section 4.3.

In everyday operations, the initial routes and the initial schedules A_k, D_k and H_k (for each subrequest r_k) are provided to the drivers, as well as the latest departure times $\overline{D}_k$. So, the drivers know how long they can wait for a late patient without violating the time constraints of the following subrequests (i.e., so that no changes must be brought to the current routes).

Since $\overline{A}_k$, $\underline{A}_k$, $\overline{D}_k$, $\underline{D}_k$ only depend on the routes and on the driver working hours, they can be calculated for every route of the initial plan from the outset. Along the day, disruptions occur, and the ensuing recovery actions may impact the routes in various ways. Therefore, the values of $\overline{A}_k, \underline{A}_k, \overline{D}_k, \underline{D}_k$ will need to be accordingly updated.

4. Real-time disruptions and recourses

This work aims at dealing with disruptions that occur along the day, by modifying the current plan when necessary. Every disruption is characterized by a *type*, a *revealed time* and an *associated recourse*.

4.1. Disruption types

The disruptions to be managed are classified into three types, to be further described in the next subsections: (1) increase of the *duration* of an appointment, (2) *reinsertion* of a request (due to a previous delay), and (3) *cancellation* of an appointment. The second type can only be met as a consequence of the first type. We do not consider the insertion of new requests in this work.

4.1.1. Increase of the duration of an appointment

Suppose that in order to fulfill subrequest r_k, a patient p is supposed to be picked up at time H_k at location l_k, but her medical appointment lasts longer than expected. If the patient terminates her appointment before time H_k, then this delay has no impact on the schedule (actually, the driver may not even observe the delay). On the other hand, if the patient is not present at time H_k, then the driver has to decide whether to wait for the patient (handle a *delay*) or to leave for the next location on the route (handle a *no-show*).

Typically, in such a case, the driver and the dispatcher have no information about the time (say, $H_k + x_k$, with x_k unknown) when patient p will actually be available for transportation. However, thanks to the information computed in Section 3.3, the driver knows that he can wait until $\overline{D}_k - s_k$ without modifying his route and still be on time to serve the following subrequests as defined by (SF2). (Observe that, contrary to intuition, there is no reason for the driver to wait until time $\overline{D}_k$: indeed, if patient p becomes available later than $\overline{D}_k - s_k$, then her service duration s_k will prevent the driver from leaving by the latest departure time $\overline{D}_k$, and hence, some of the subsequent subrequests will be infeasible.) If patient p shows up at time $H_k + x_k \leq \overline{D}_k - s_k$, therefore, she can get on board and her delay has no consequences for the remaining sequence of subrequests on route ρ_j. Nonetheless, the schedule $A_{k+1}, H_{k+1}, D_{k+1}, \ldots, A_{K_j}, H_{K_j}, D_{K_j}$ of the subsequent subrequests must be adapted. The driver sends a notification to the dispatching office when leaving location l_k at time D_k for this purpose.

If patient p has not arrived at time $\overline{D}_k - s_k$, yet, then the driver will not be able to take care of this patient without being late for at least one of the following subrequests of his route. Since the magnitude x_k of the patient's delay is not known, it is impossible to predict its impact on the next subrequests $(r_{k+1}, \ldots, r_{K_j})$. Therefore, in this case, we adopt the dispatching rule that the driver leaves at time $\overline{D}_k - s_k$ without patient p, who is *temporarily* considered as a no-show. The driver notifies the dispatching office of

this action at time $\overline{D}_k - s_k$. Both the pickup subrequest and the associated delivery subrequest of patient p are thus removed from the route at time $\overline{D}_k - s_k$, with the implications that the remaining subrequests of the route have to be rescheduled, and that the decisions regarding the inbound trip of patient p are postponed. This pair of postponed subrerequests of patient p will reappear in the future, once her appointment will actually be over. We next turn a discussion of this event.

4.1.2. Reinsertion of a request

When the appointment of a patient p finishes later than expected and the driver has already left the pickup location l_k, the patient informs the dispatching office of the company as soon as she is ready for her inbound trip, say, at time $H_k + x_k > \overline{D}_k - s_k$. This leads to a shift of the pickup and delivery time windows of patient p by $x_k + (H_k - a_k)$ time units. The dispatching office then tries (at time $H_k + x_k$) to reinsert this pair of subrequests in the route of one of the vehicles. The main goal is to find a feasible reinsertion so as to avoid calling upon external companies to take charge of the patient. A single appointment duration increase can thus lead to two actions (removal and reinsertion of two subrequests) at two distinct moments. To the best of our knowledge, this situation has not been considered in the DARP literature. The reinsertion recourse procedures are described in Section 4.3.

4.1.3. Cancellation of a request

Appointments and requests may be canceled for various reasons. For instance, a patient may be too weak to travel, or the machine used for a medical treatment may be unexpectedly unavailable. Such events result in a cancellation of the appointment and hence, in the cancellation of the patient's outbound and inbound journeys. In other cases, the patient may decide to cancel one of her trips only (e.g., because a relative takes care of it). All such situations affect one or several pairs of subrequests, namely, both the pickup subrequest and the associated delivery subrequest must be removed from the route.

4.2. Revealed time

The *revealed time* of a disruption is, by definition, the time at which the dispatching office becomes aware of the disruption. Note that the revealed time may differ from the time at which the disruption has actually started. The revealed time is used as a time stamp for the simulation clock in the discrete event algorithm in Section 5.3.

In case of an increased appointment duration (associated with pickup subrequest r_k for patient p), the revealed time may be the moment $H_k + x_k \leq \overline{D}_k - s_k$ when patient p boards the vehicle that was initially assigned to her, if the vehicle kept waiting long enough, or otherwise it is the moment $\overline{D}_k - s_k$ when the driver leaves l_k without patient p. For a request reinsertion, the revealed time is the moment $H_k + x_k > \overline{D}_k - s_k$ at which patient p calls the dispatching office to inform them that she is available for her inbound trip. For a request cancellation, the revealed time is the moment at which the dispatching office is informed of the cancellation.

4.3. Associated recourses

In order to deal with a disruption affecting subrequest r_k, an action has to be undertaken. This action, called a *recourse*, is determined by an appropriate algorithm (akin to a "recovery algorithm" in the framework of Liebchen et al., 2009). The recourse to be applied depends on the type of the disruption and on the current solution.

A. [No impact delay] As discussed earlier, if the end of an appointment associated with r_k is postponed but the patient still comes out before her planned pickup time H_k, then no change is notified and the planning does not require any modification. This situation is labeled as a *no impact delay* in the sequel.
B. [Delay] If the appointment ends at time $H_k + x_k$ $(x_k \geq 0)$, but before the limit $\overline{D}_k - s_k$, then the route is not modified, only the schedule is. The time windows of the pickup r_k and the associated delivery subrequests are shifted further in time (by $x_k + (H_k - a_k)$ time units), and consequently, the earliest and latest arrival and departure times of the subsequent subrequests $r_{k+1}, \ldots, r_{K_j}$ are updated, as well as $A_{k+1}, H_{k+1}, D_{k+1}, \ldots, A_{K_j}, H_{K_j}, D_{K_j}$. This recourse is labeled as a *delay*.
C. [Postponed action] If the appointment ends too late ($H_k + x_k > \overline{D}_k - s_k$), then the route must be modified by temporarily canceling the pickup and delivery of this patient. This recourse is labeled as a *postponed action* in the sequel. An additional disruption corresponding to the reinsertion of these subrequests will be later revealed at the (unpredictable) time $H_k + x_k$, and will be handled by the next type of recourse.
D. [Reinsertion] When an action is postponed as in recourse C, the delayed patient submits a request for her inbound trip at time $H_k + x_k$. As a result, two subrequests (pickup and delivery) need to be reinserted in a route as explained in Section 4.1.2. This recourse is labeled as a *reinsertion*.
E. [Definitive cancellation] Due to a cancellation by the patient or by the health center, a request may have to be removed from the plan as explained in Section 4.1.3. This recourse is labeled as a *definitive cancellation* in the following sections:

The most difficult recourse to handle is the reinsertion of a request. The reinsertion procedure must take care of two subrequests: the patient needs to be picked up at a given location and then delivered at another location. These two subrequests, say r^+ for the pickup and r^- for the delivery, should be both reinserted in the same route to take care of the patient's inbound trip. As noted by Berbeglia, Pesant, and Rousseau (2011), checking the feasibility of a DARP instance is NP-complete, as a consequence of the NP-completeness of the traveling salesman problem with time windows (Savelsbergh, 1985; see also Berbeglia, Pesant, & Rousseau, 2012 for related results). Berbeglia et al. (2011) provide a constraint programming algorithm to assess the feasibility of inserting a new request in a feasible plan. Their algorithm either returns a feasible plan that includes the new request without modifying the existing routes, or proves that there is no such feasible plan. While this algorithm answers some of the questions arising in our setting, its implementation is quite complex and does not offer the flexibility to "fix" a route when the reinsertion fails. Moreover, its running time is relatively high (at least, when compared with the running times of our algorithms). So, we rather choose to introduce four distinct simple *reinsertion operators* in order to successfully reinsert each pair of subrequests.

4.3.1. Reinsertion operator O1: pure reinsertion

The first operator, O1, is a pure reinsertion operator. First, O1 attempts to insert the pickup subrequest r^+ in every position of every route of the current solution which is compatible with the subrequest time window. The insertion is feasible if the conditions (1) are satisfied for all subrequests r_k in the route where the insertion is attempted. If one or several insertions are feasible for r^+, then for every possible choice, the operator tries to insert the associated delivery r^- in the same route as the pickup. Each feasible delivery insertion leads to a feasible solution for the request reinsertion. When several feasible solutions still exist, the operator O1

Table 4
Allowed extensions of a subrequest time window $[a, b]$ ($\eta, \zeta > 0$).

	Pickup	Delivery
Outbound	$[a-\eta, b]$	$[a-\eta, b]$
Inbound	$[a, b+\zeta]$	$[a, b+\zeta]$

selects the one that maximizes the average slack (as defined in (2)) over all subrequests (*average slack criterion*).

4.3.2. Reinsertion operator O2: destroy-and-repair

If no solution for the two unassigned subrequests r^+, r^- can be found with the first operator, then a second reinsertion operator, O2, based on a destroy-and-repair strategy is considered. In every route ρ_j independently, O2 first selects the set of subrequests "competing" either with the unassigned pickup r^+ or with the unassigned delivery r^-: here, a subrequest r_{k^*} is said to be "competing" with $r^\pm$ with time window $[a^\pm, b^\pm]$ if $[a_{k^*}, b_{k^*}] \cap [a^\pm, b^\pm]$ is non empty. Say, e.g., that subrequest r_{k^*} is a pickup (respectively, delivery) subrequest competing with either r^+ or r^-. Let r_{i^*} be the delivery (respectively, pickup) subrequest paired with r_{k^*}. Then, r_{k^*} and r_{i^*} are removed from their current route ρ_j, and the operator O1 is used to reinsert the pair (r_{k^*}, r_{i^*}) either later in the same route or in another route. If the reinsertion of the pair (r_{k^*}, r_{i^*}) succeeds, then O1 attempts to reinsert the pair (r^+, r^-) in the route ρ_j. If several feasible reinsertions exist with this destroy-and-repair operator O2, then the solution that maximizes the average slack over all subrequests is selected.

The two reinsertion operators O1, O2 are hierarchically applied since we target feasible reinsertions which least affect the current routes, so as to avoid confusion. It may happen, however, that no feasible reinsertion can be found for a request with either of the two operators O1 and O2. In this case, we allow slight relaxations of the problem, as described next.

4.3.3. Reinsertion operator O3: relaxed time windows

The third reinsertion operator, O3, is based on a relaxation of the subrequest time windows. This relaxation must be consistent: all patients should still be on time for their appointments, therefore only the beginning of the pickup time window of an outbound trip (to the health center) can be shifted earlier in time (a patient can be loaded earlier than initially). As for the inbound trip (back home), a patient cannot be picked up before the end of her appointment, but the maximum journey duration may be extended. Consequently, we allow the extensions of time windows displayed in Table 4. The parameters η and ζ control the magnitude of the extensions: the larger η or ζ, the more likely a successful reinsertion. (The initial parameterization of the subrequest time windows will be discussed in Section 5.2.)

In practice, when O1 and O2 fail, we try again to apply O1 while allowing the extension of the time windows of the subrequests to be reinserted (i.e., r^+ and r^-), as well as those of all subrequests in the route where the reinsertion takes place. If several feasible insertions are found with this third operator O3, the solution minimizing the total extension with respect to the initial subrequest time windows is selected. Ties are broken using the average slack criterion.

Note that (contrary to what is assumed by Schilde, Doerner, & Hartl, 2011, for example) the time windows can be extended only within limits defined by η and ζ, so that O3 is not guaranteed to find feasible reinsertions.

4.3.4. Reinsertion operator O4: extended working hours

If the reinsertion operator O3 is not sufficient to find a feasible solution, we attempt again to apply O1 while allowing to extend the drivers' working hours in addition to the subrequest time windows. This means that for every route j, β_j is increased to $\beta_j + \xi$ with $\xi > 0$. As in the case of O3, the value of ξ controls the magnitude of the extension: the larger ξ, the more likely a successful reinsertion. (Note, however, that the extension is never applied to a driver who is already back to the final depot.) If several feasible solutions are found with this reinsertion procedure, the solution minimizing the total driver overtime is selected. Ties are broken based on the total extension of the subrequest time windows and, if necessary, on the average slack criterion. This defines a fourth reinsertion operator O4.

4.3.5. Buffered reinsertions and failures

If none of the four operators O1, O2, O3, O4 can identify a feasible reinsertion at the time when a disruption is revealed, then the unassigned request is stored in a buffer list. The requests in this buffer list receive another chance to be reinserted in the planning during a predetermined buffer period after their revealed time, whenever a new disruption occurs. (Indeed, a reinsertion is likely to be successful only if the schedule changes.) The buffer contains a list of unassigned requests sorted by chronological revealed times. When a new disruption is successfully handled, each request of the buffer list is examined in turn: if the request has been in the buffer list for less than the maximum authorized period, then a reinsertion for this request is attempted; if the buffer period is over, the request is removed from the buffer list, meaning that this request will never be reinserted and thus, that the patient has to be taken back home by other means (taxi, volunteer driver, etc.). Such a lost request is called a *failure*. Note that some failures may be due to the limitations of our procedures, while others may be unavoidable: in particular, if a patient is ready for pick-up after the end of the workday of the last driver, then a failure necessarily occurs.

5. Computational experiment settings

In this section, we describe the set of instances, the parameters related to the specific application setting, and the scenarios generated for our computational experiments. All time-related quantities are expressed in minutes, for simplicity.

5.1. Instances

As mentioned in Section 2, most of the scientific work on the DARP is motivated by practical problems and therefore, the algorithms are mainly tested on case-specific real-life instances (Ho et al., 2018; Molenbruch et al., 2017). Two sets of artificial benchmark data have been proposed to perform comparisons between developed algorithms (Cordeau, 2006; Cordeau & Laporte, 2003; Ropke, Cordeau, & Laporte, 2007). However, none of these two data sets presents disruptions enabling us to compare our algorithm with other previously published techniques. For this reason, we chose to use an existing collection of realistic instances (Thomas, Cappart, Schaus, & Rousseau, 2018) based on the case of our non-profit partner and to create disruption scenarios as explained in Section 5.3. The original (deterministic) DARP instances are available online at http://www.csplib.org/Problems/prob082. The first ten instances of each of the three categories (easy, medium and hard) are selected, and we use as initial plans the solutions computed by Thomas et al. (2018). This yields a set of 30 distinct instances, where each instance is associated with an initial collection of routes and schedules. The total number of requests in the instances varies from 14 to 274, with a mean equal to 101.93, and the number of inbound requests varies from 8 to 139, with a mean equal to 51.53. The number of routes is between 2 and 16.

Table 5
Construction of the subrequest time windows for a given patient.

	Start of the pickup		Start of the delivery	
	T.w. beginning	T.w. end	T.w. beginning	T.w. end
Outbound	$[at - f,$	$at - 2s - t_{l_1,l_2}]$	$[at - f + s + t_{l_1,l_2},$	$at - s]$
Inbound	$[at + d,$	$at + d + m - s - t_{l_2,l_3}]$	$[at + d + s + t_{l_2,l_3},$	$at + d + m]$

5.2. Case details

In Section 3, the subrequest time windows were described in a generic framework. We now provide more details about their parametrization in our specific application. Of course, all these values could easily be modified to account for different real-world environments.

As explained already, each patient may have two types of requests: one outbound request (taking the patient to the medical appointment) and one inbound request (taking the patient back home after the appointment). Each type of request has its own constraints. For instance, the inbound trip may have a maximum duration, say, m minutes. This is typically the case after a tiring medical appointment, such as a dialysis. More details about the value of m will be provided below. As for the outbound request, the patient should reach the appointment location on time, but can arrive earlier. The appointment start time (at) and the projected appointment duration (d) are provided. When a patient asks for outbound transportation, the earliest pickup time is automatically set f minutes before the appointment time at. According to Paquette et al. (2009), in the DARP literature, the subrequest time windows are usually 15 minutes long and the maximum trip time is set to 90 minutes. In our application, f is set to 30 minutes for all the patients, due to the very local area served by the organization. As a reminder, s is the service duration, that is the time required for loading or unloading the patient in a vehicle. For the sake of simplicity, here the service durations are assumed to be equal for loading and unloading but they could easily be set to different values.

Time windows for both pairs of subrequests can then be calculated on the basis of the previous parameters: Table 5 displays the feasible time windows for the start of each pickup or delivery subrequest. The patient is assumed to be picked up at location l_1 and delivered at location l_2 on the outbound trip, and brought back from location l_2 to location l_3 at the end of the inbound trip.

We only define the maximum time limit m for inbound requests. It is considered as the maximum journey duration, that is, from the moment $(at + d)$ when the patient is available after her appointment until her delivery at home, including the possible waiting time before pickup, the ride time and the service time at the delivery location. In our experiments, the maximum journey duration is proportional to the direct journey duration, that is, $t_{l_2,l_3} + 2s$, unless this duration is very small. Thus, similarly to Häll, Högberg, and Lundgren (2012), we define m as follows:

$$m = \begin{cases} \theta(t_{l_2,l_3} + 2s) + A & \text{if } (t_{l_2,l_3} + 2s) < M \\ \theta(t_{l_2,l_3} + 2s) & \text{if } (t_{l_2,l_3} + 2s) \geq M. \end{cases} \tag{3}$$

The values of the parameters θ, M and A are set to 2, 15 minutes and 30 minutes, respectively. The constraint on the maximum journey duration is ensured through the definition of the subrequest time windows from Table 5.

Consider now the possibility of extending some subrequest time windows, as explained in Section 4.3. The values of parameters η and ζ in Table 4 are set so as to allow an increase of the subrequest time windows by 50%. As for the drivers' working hours, an extension of up to 120 minutes is allowed, i.e., $\xi = 120$.

Finally, the buffer period (see Section 4.3.5) is set to 15 minutes in order to prevent the patients from waiting too long for an answer from the company.

5.3. Simulated disruptions

In order to create different situations, we create random scenarios of real-time disruptions. For a given instance, a scenario is defined as a set of disruptions that apply to the associated plan. It can be seen as the sequence of events occurring on a given day. The random elements are the following.

Increase of appointment duration. This type of disruption only concerns the inbound request of a patient. A Gamma random variable is used to define the actual duration of the patient appointment. The parameters of the distribution are chosen so that the mean is equal to the (deterministic) appointment duration (say, d) used in the original instance, while the standard deviation is equal to δd, for a given parameter $\delta \in [0, 1]$. If the value generated for the Gamma variable is γ, then the actual duration of the appointment is set equal to $\max\{d, \gamma\}$ in the scenario under consideration. So, we assume that an appointment cannot be shorter than initially planned: this assumption reflects the fact that shorter durations rarely occur in reality, and that, even when they occur, the patient is unlikely to communicate the information to the dispatching center. Therefore, we only consider disruptions leading to appointment durations that are larger than expected and that follow a truncated Gamma distribution.

Request cancellation. Cancellations can apply to only one request (either outbound or inbound, if the patient finds another transportation means), or to a pair of requests (for example, in case of an appointment cancellation). In order to create the scenarios, for each patient, a Bernoulli random variable is first generated to determine whether the appointment is maintained (with probability $1 - p$) or is canceled (with probability p). If the appointment is maintained, then another Bernoulli trial is used for each existing request (outbound and inbound), independently, with the same cancellation probability p.

Cancellations are typically known in advance of the request planned time, and thus, the moment at which the dispatching office becomes aware of the cancellations must also be determined. If the appointment or a single journey is canceled, then a Gamma random variable is used to determine how long in advance the cancellation is announced. On average, the disruption is announced 60 minutes before the pickup time window, with a standard deviation equal to 15 minutes.

Set of scenarios. For every instance and for certain values of (p, δ), a set of 50 random scenarios is generated. The pairs (p, δ) that we consider are

- $(p, \delta) = (0.05, 0.25)$,
- $(p, \delta) \in \{(0.05, 0.01), (0.05, 0.10), (0.05, 0.40)\}$, with $p = 0.05$ fixed, and
- $(p, \delta) \in \{(0.01, 0.25), (0.1, 0.25), (0.2, 0.25), (0.4, 0.25)\}$, with $\delta = 0.25$ fixed.

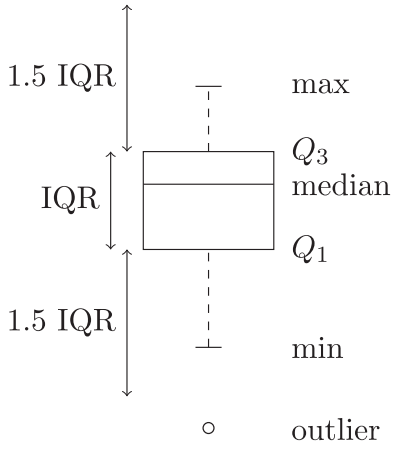


Fig. 2. Data represented by a boxplot.

Note that each of the reference values $p = 0.05$ and $\delta = 0.25$ is kept fixed for comparison when the second parameter varies. The smallest and largest values of p and δ are unlikely to be observed in practice, but will allow us to validate the behavior of our procedures under rather "extreme" conditions.

In total, this produces a collection of 12 000 scenarios (30 instances $\times$ 8 pairs of parameters $\times$ 50 scenarios) to be handled by the procedures described in Section 4.

The disruptions in each scenario are successively handled by the recourse procedures described in Section 4.3 at the time when they are revealed, in a discrete event simulation fashion, as they would be by the dispatching office in a real-life application. A schematic description of the simulation process is summarized by Algorithm 1.

Algorithm 1 Discrete event algorithm.

```
while the day is not over do
    Wait for a disruption dis revealed at time t_dis
    Generate the recourse for dis
    if recourse(dis) is postponed action then
        Create later reinsertion disruption
    end if
    Deal with generated recourse
end while
```

6. Results

The aim of the computational experiments is to establish the effectiveness of the algorithm approach in handling the real-time disruptions, but also to evaluate the impact of the disruptions on the quality of the solutions.

The algorithm was implemented in Java. All calculations were performed on a personal laptop with 16.0 gigabytes RAM and an Intel Core i5-7300U (2.6 gigahertz) processor running 64-bit Windows 10 Pro. The algorithm was executed on the 12,000 scenarios described in the previous section. Each run took less than two seconds for a complete scenario. This speed confirms that the proposed procedures are in line with the purpose of the work and are appropriate for use in real time.

In the sequel, several solution quality measures and algorithmic performance measures are collected for different values of p and δ. We rely on *boxplots* (Tukey, 1977) to visualize these observations. A boxplot, as illustrated in Fig. 2, is a graphical representation of a sample of numerical data through their quartiles. The box represents the 50% of data lying between the first and third quartiles (i.e., the interquartile range $IQR = Q_3 - Q_1$), the band inside the box indicates the median. The lower whisker end (min) is the smallest observation within 1.5 IQR of the lower quartile Q_1, and the upper whisker end (max) is the largest observation within 1.5 IQR of the upper quartile Q_3. Any data point that is not included between the whiskers is viewed as an outlier and is plotted as a small circle. Boxplots enable the reader, with a simple glance, to perceive the degree of dispersion and skewness in the data.

6.1. Quality measures

The quality of a solution relates to two dimensions. First, the objective function of the initial plan creation, that is, satisfying the maximum number of requests, must be taken into account, as it most directly relates to the mission of the organization. In the present case, it means that most, if not all of the requests initially placed in the routes should be served and thus, that the number of reinsertions should be maximized. Second, patient satisfaction should be optimized as well. Reinserting certain requests may impact the service level, as measured through the excess journey duration ratio. Some reinsertions can come at the cost of time constraints violation and driver overtime. These different dimensions are analyzed next.

6.1.1. Failed scenarios and unassigned requests

The main focus of this work is on managing all the disruptions, that is, on taking care of all the patient requests selected in the initial plan. The only disruption potentially leading to an infeasibility is the reinsertion of a request following a postponed action (recourse D in Section 4.3), since delays and cancellations can always be tackled. Note that reinsertions may be especially troublesome in our framework, because the selection of requests performed in order to generate the initial plan aims at maximizing the number of patients served among a larger demand pool (Thomas et al., 2018) and hence, the associated schedules are likely to be tight.

A first key performance indicator (KPI), denoted KPI_1, is provided by the proportion of failed scenarios, that is, the number of scenarios with at least one unsuccessful reinsertion (= failure, see Section 4.3.5), divided by the total number of scenarios (i.e., 50 in these simulations) for a given instance. A failed scenario means that, over the whole day, there is at least one unserved inbound request during the drivers' working hours (unavoidable failures due to a patient delay extending beyond the latest driver's end of the day are not taken into consideration for this KPI). So, for a given instance, KPI_1 is defined as

$$KPI_1 = \frac{\#\text{failed scenarios}}{\#\text{scenarios}}.$$

In Fig. 3, each boxplot corresponds to a pair of parameters (p, δ). Within a boxplot, each observation corresponds to the value of KPI_1 associated with one instance. Therefore, each boxplot in Fig. 3 holds 30 observations.

On the left-hand side of Fig. 3, one can see that when δ is small, that is, when the delays are rather small, then the proportion of failed scenarios is small as well. When the magnitude of the delays increases (larger values of δ), however, the ratio of failed scenarios tends to increase as well. For $\delta = 0.4$, 75% (below the third quartile) of the instances have fewer than 40% of failed scenarios, but some instances can reach 74% of failed scenarios.

As shown on the right-hand side of Fig. 3, there is no obvious trend for the values of p between 0.01 and 0.10. Here again, some instances may be associated with a rather high percentage of failed scenarios, even though 75% of the instances show fewer than 50% of failed scenarios. For larger values of p, the number of failed scenarios decreases. This behavior may be expected: when more appointments are canceled, there is more slack in the schedules and

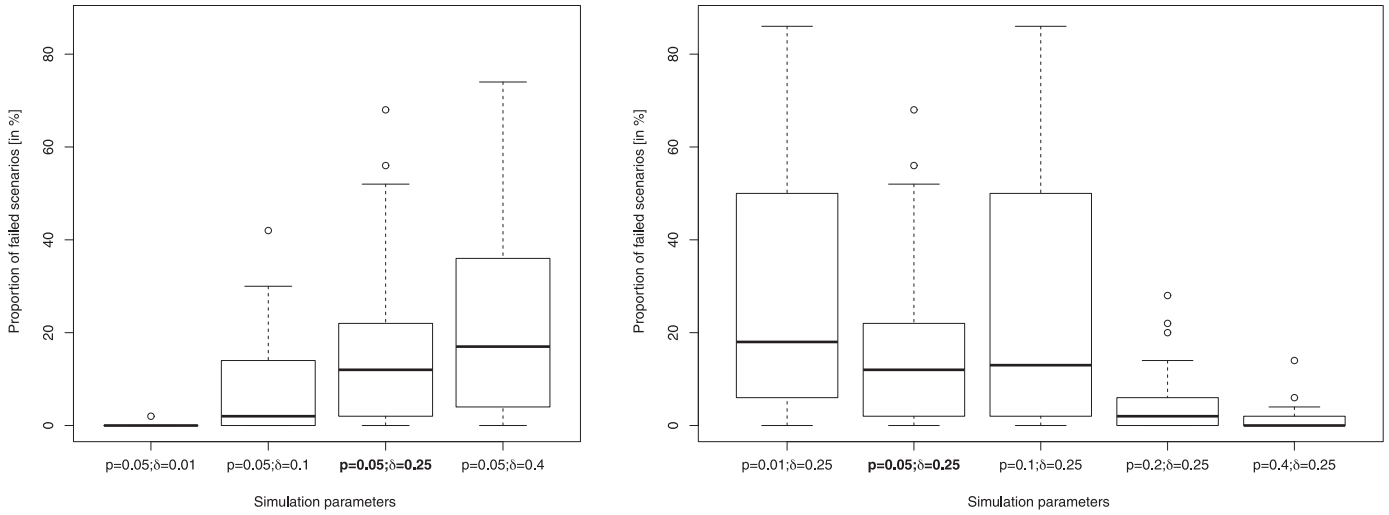


Fig. 3. Proportion of failed scenarios for different values of δ (left) and p (right).

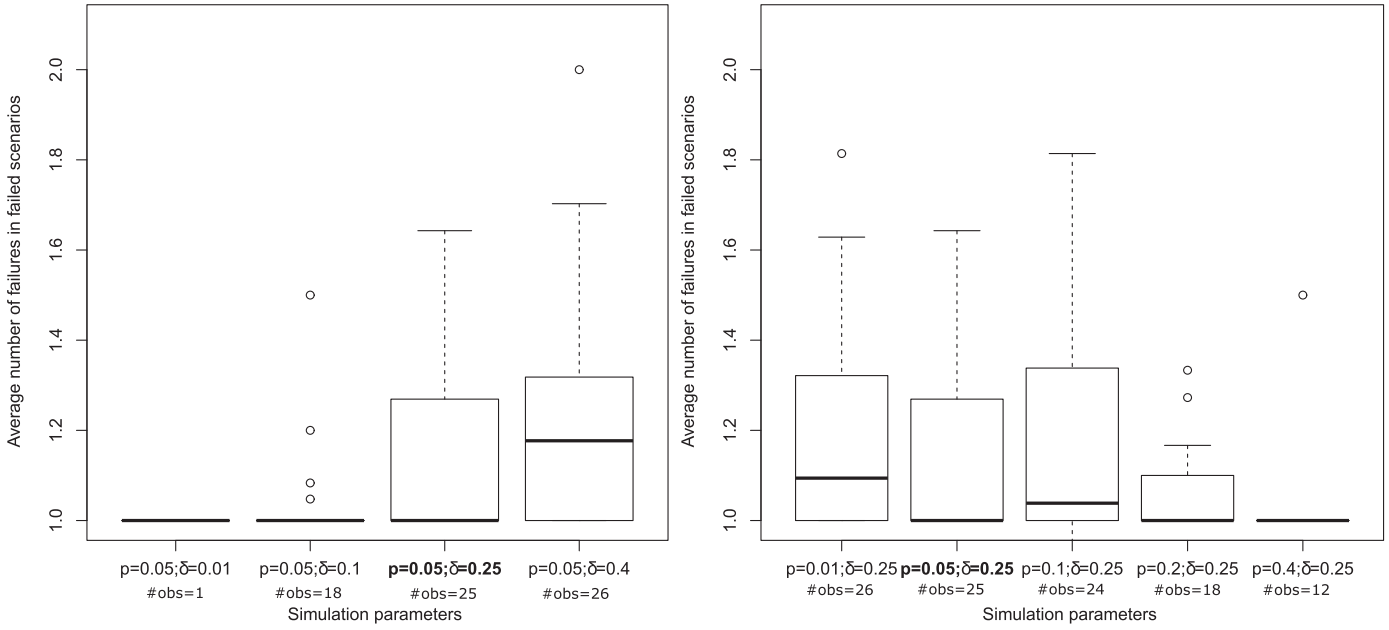


Fig. 4. Average number of failures for different values of δ (left) and p (right).

reinsertions are more likely to become feasible, thus resulting in fewer failed scenarios.

These comments regarding KPI_1 should be interpreted with caution since a failed scenario may either contain a single unserved request, or several ones. In risk management parlance, we would say that KPI_1 focuses on the *frequency* of failures, but not on their *severity*. In order to take the latter criterion into account, we introduce a second indicator, KPI_2, which represents the average number of unserved requests among the failed scenarios. More precisely, for a given instance with at least one failed scenario, KPI_2 is defined as:

$$KPI_2 = \frac{\sum_{\text{failed scenarios } s} \#\text{failures in } s}{\#\text{failed scenarios}}.$$

Clearly, KPI_2 cannot be smaller than 1. The boxplots in Fig. 4 display the value of KP_2 for all instances with at least one failed scenario. The number of observations is indicated for each pair of parameters (p, δ).

The left-hand part of Fig. 4 shows that if the appointments last just a little longer than expected (small values of δ), there is on average at most one lost request per scenario. For larger delays, KPI_2 tends to increase, but never exceeds two unserved requests per scenario. Similar conclusions apply to the right-hand side of Fig. 4. So, considering the severity indicator KPI_2 suggests that, from a practical point of view, the performance of our simple recourse procedures is quite satisfactory, as the number of lost requests is extremely small. (Recall that there are, on average, more than 100 requests per instance.)

Note again that the number of unserved requests tends to decrease with the value of δ and to increase with the value of p. This corroborates the observations already made about KPI_1, and confirms that the algorithm behaves as expected.

6.1.2. Excess journey duration ratio

A relevant indicator for patient satisfaction is the excess journey duration ratio (EJDR), defined as the ratio of the actual journey duration, including service duration and waiting time before pickup,

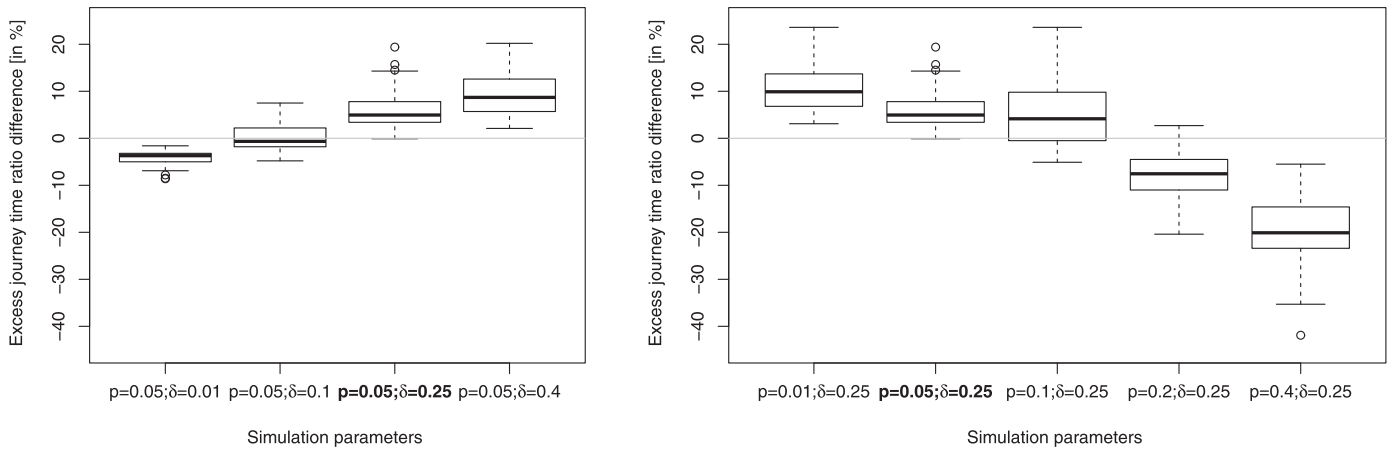


Fig. 5. Average EJDR difference for different values of δ (left) and p (right).

over the direct journey duration (Paquette et al., 2009). More precisely, for a pair $r = (r_{k_1}, r_{k_2})$ of associated subrequests, where r_{k_1} is the patient pickup and r_{k_2} the patient delivery, with $k_1 < k_2$,

$$\text{EJDR}(r) = \frac{\overbrace{(H_{k_2} + s_{k_2})}^{\text{end of delivery}} - \overbrace{a_{k_1}}^{\text{start of availability}}}{\underbrace{s_{k_1} + t_{k_1,k_2} + s_{k_2}}_{\text{direct journey duration}}}.$$

The closer to 1 this ratio, the more satisfied the patient. (Note that our policy to set the values of A_k, D_k, H_k to the earliest possible times, as explained in Section 3.3, tends to minimize the value of EJDR.)

In our experimental framework, the aim is to measure the difference between the EJDR of the initial solution and the EJDR in the final solution (that is, considering the shifted time windows and the A_k, H_k, D_k that actually happened a posteriori), in order to quantify the changes due to the disruptions. Therefore, for each pair r of subrequests in the final solution, we compute the EJDR difference, $\Delta\text{EJDR}(r)$:

$$\begin{aligned}\Delta\text{EJDR}(r) &= \text{EJDR}_{\text{final}}(r) - \text{EJDR}_{\text{initial}}(r)\\ &= \frac{\big((H_{k_2} + s_{k_2}) - a_{k_1}\big)_{\text{final}} - \big((H_{k_2} + s_{k_2}) - a_{k_1}\big)_{\text{initial}}}{s_{k_1} + t_{k_1,k_2} + s_{k_2}}\end{aligned}$$

If the difference is positive, it means that for this request, the journey is longer than initially planned. If the difference is negative, then the journey is shorter than initially planned, for instance because a detour initially planned for another request has been removed.

For a scenario s, the average EJDR difference is computed over all requests (i.e., pairs of associated subrequests) occurring in the final solution sol(s) obtained for this scenario. For a given instance, the average (over 50 scenarios) of the average EJDR differences (over all requests) is considered as a third KPI:

$$KPI_3 = \frac{1}{\#\text{scenarios}} \sum_{\text{scenarios } s} \times \left(\frac{1}{\#\text{requests in sol}(s)} \sum_{\text{requests } r \text{ in sol}(s)} \Delta\text{EJDR}(r) \right).$$

Each instance has a value for KPI_3 and each boxplot in Fig. 5 thus holds 30 observations.

On the left-hand side of Fig. 5, one can see that for small delays, KPI_3 is negative, probably because of the nonzero value of p. Indeed, the delays are small ($\delta = 0.01$) and several requests are canceled ($p = 0.05$), the drivers have thus less subrequests to serve and some detours (from the patients' point of view) are likely to be canceled. For larger values of δ, KPI_3 increases and becomes positive, meaning that the patients wait or stay on board for a longer time. This observation is as expected: if delays grow larger, the driver is likely to wait more frequently for late patients, possibly with other patients on board. We will also see in Section 6.2 that the number of reinserted requests increases with the magnitude of the delays; then, the patients are more likely to wait, and possible detours are also more likely to be introduced in the route to serve the delayed patients.

On the right-hand side of Fig. 5, one observes that the larger p, the smaller KPI_3. Indeed, cancellations tend to decrease the number of detours in the routes; moreover, the drivers are more likely to be present when the patients terminate their appointments, thus creating less driving time and waiting time for the patients.

6.1.3. Time window extensions

As explained in Section 4.3, when trying to reinsert a request, the third operator O3 is based on a relaxation of the time window constraints. Once again, this extension may impact the patient satisfaction: the larger the extension, the worse the service quality. Like for the reinsertion analysis here above, we will first analyze the frequency of these time window extensions using an indicator KPI_4, and next their severity through an indicator KPI_5.

Let $\mathcal{S}$ be the set of all scenarios (in this case, of the 50 scenarios) for a given instance. For a scenario $s \in \mathcal{S}$, let $TW^{\text{ext}}(s)$ denote the number of subrequests whose time window is extended in the final solution sol(s) for this scenario. We compute the proportion of extended time windows over all the subrequests included in sol(s). For each instance, KPI_4 is defined as the average of these proportions over all the scenarios in $\mathcal{S}$:

$$KPI_4 = \frac{1}{\#\mathcal{S}} \sum_{\text{scenarioss} \in \mathcal{S}} \left(\frac{TW^{\text{ext}}(s)}{\#\text{subrequests in sol}(s)} \right)$$

Each instance has a value for KPI_4 and hence, each boxplot of Fig. 6 holds 30 observations.

Fig. 6 shows that the proportion of extended time windows increases with the value of δ and decreases with the value of p: this confirms that the third operator is sometimes useful in order to find a feasible solution when the delays are large and when the number of cancellations is small. Most importantly, however, the average proportion of extended time windows never exceeds 1.5% for any instance, whatever the parameter values. From this perspective, the quality of the service to the patients is rarely degraded.

We now focus on the severity of the extensions. For this purpose, we consider only the scenarios with at least one extended

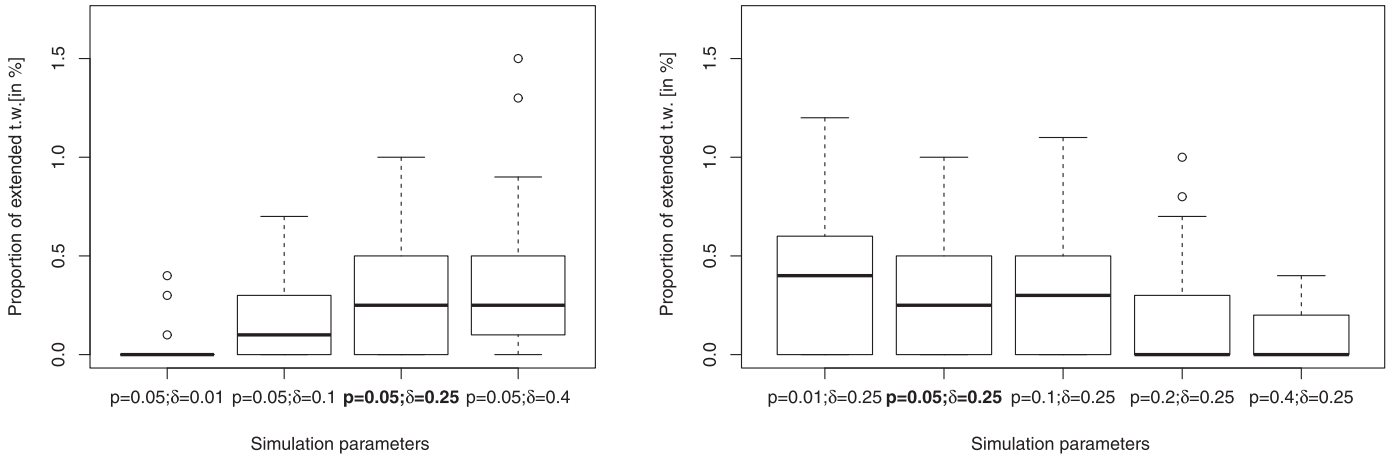


Fig. 6. Average proportion of extended time windows for different values of δ (left) and p (right).

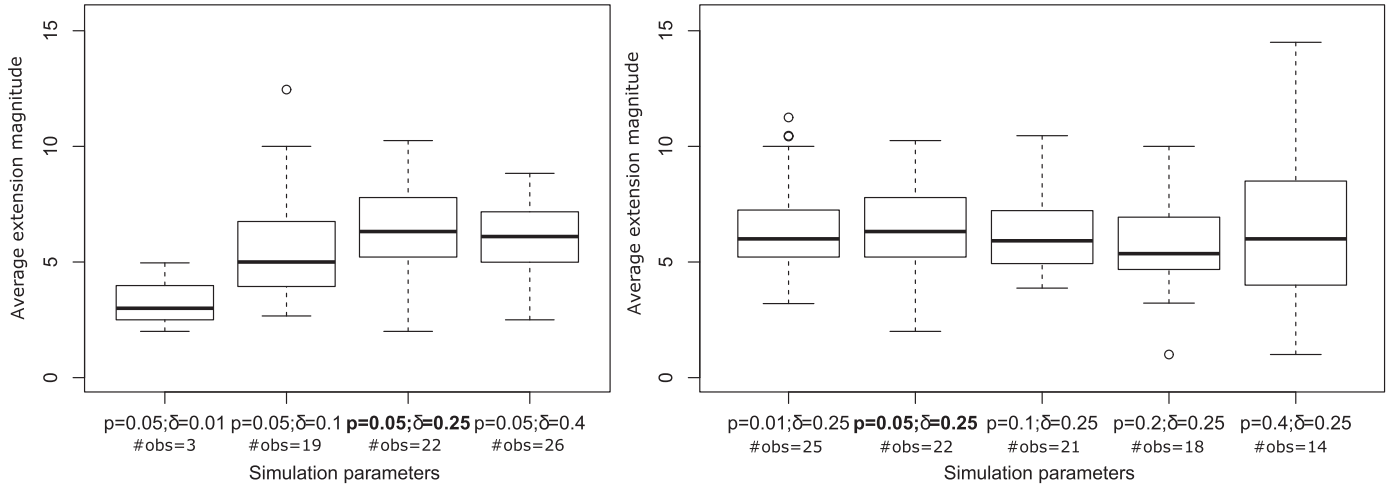


Fig. 7. Average time window extension in minutes for different values of δ (left) and p (right).

time window. For a given instance, let $\mathcal{S}^{TW}$ be the subset of scenarios with at least one extended time window, that is, $\mathcal{S}^{TW} = \{s \in \mathcal{S} : TW^{\text{ext}}(s) > 0\}$. The magnitude of an extended time window is defined as $\max\{H_k - b_k, a_k - H_k\}$. For a given instance, KPI_5 is then defined as the average, over all scenarios in $\mathcal{S}^{TW}$, of the average extension magnitude over all the extended time windows:

$$KPI_5 = \frac{1}{\#\mathcal{S}^{TW}} \sum_{\text{scenarios } s \in \mathcal{S}^{TW}} \times \left(\frac{1}{TW^{\text{ext}}(s)} \sum_{\text{extended t.w. } k \text{ in sol}(s)} \max\{H_k - b_k, a_k - H_k\} \right).$$

If the instance has no scenario with at least one extended time window (meaning that $\mathcal{S}^{TW}$ is empty), the instance is not considered in the boxplot in Fig. 7. The number of observations in each boxplot represented is indicated for each pair of parameters (p, δ).

Fig. 7 does not show any clear trend for the extension magnitude. But we observe that the average extension rarely exceeds 10 minutes, which is a very reasonable value in terms of service quality, especially in view of the low frequency of the recourse to extensions (cf. Fig. 6). Note also that KPI_5 is a rather pessimistic indicator: indeed, it disregards the case of those patients who get home earlier than initially planned and hence, whose satisfaction increases due to the occurrence of the disruptions and to our way of handling them.

6.1.4. Drivers' overtime

The fourth operator O4 designed for reinserting a request is based on the extension of the working hours of the drivers. This recourse affects the drivers' satisfaction and entails additional costs for the company. Here again, the analysis is split into two parts: we first analyze the frequency of driving hours extensions, then the severity of the extensions.

Let $J(s)$ denote the number of routes in the final solution sol(s) for scenario s. A route ρ_j of sol(s) is in overtime if the arrival time A_{K_j+1} at the final depot exceeds the normal end of the working day for the driver, i.e., if $A_{K_j+1} \in]\beta_j, \beta_j + \xi]$. Let $R^{\text{over}}(s)$ denote the number of routes in overtime. For each instance, KPI_6 is the average, over all the scenarios in $\mathcal{S}$, of the proportion of routes in overtime over the total number of routes:

$$KPI_6 = \frac{1}{\#\mathcal{S}} \sum_{\text{scenarios } s \in \mathcal{S}} \left(\frac{R^{\text{over}}(s)}{J(s)} \right)$$

This KPI is well-defined for each instance, and hence each boxplot in Fig. 8 holds 30 observations.

Fig. 8 suggests that few drivers have to work overtime (fewer than 5 or 6% of them on average for most instances). Thus, even though this recourse is helpful in order to find feasible solutions when facing large delays, it does not lead to an excessive occurrence of overtime. The behavior of KPI_6 in relation with δ and p is, once again, in line with our intuition.

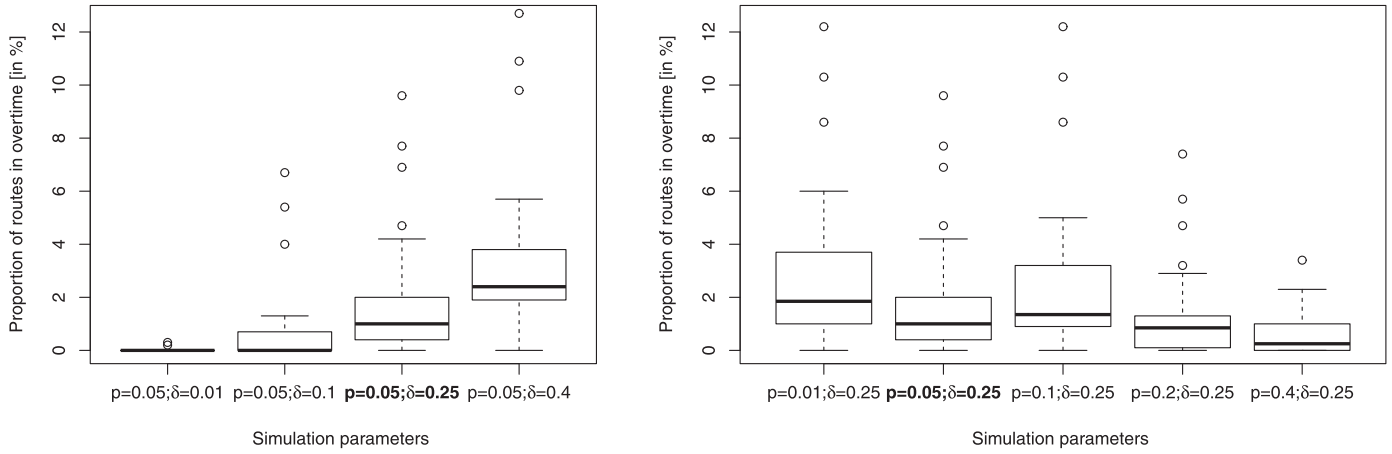


Fig. 8. Average proportion of routes in overtime for different values of δ (left) and p (right).

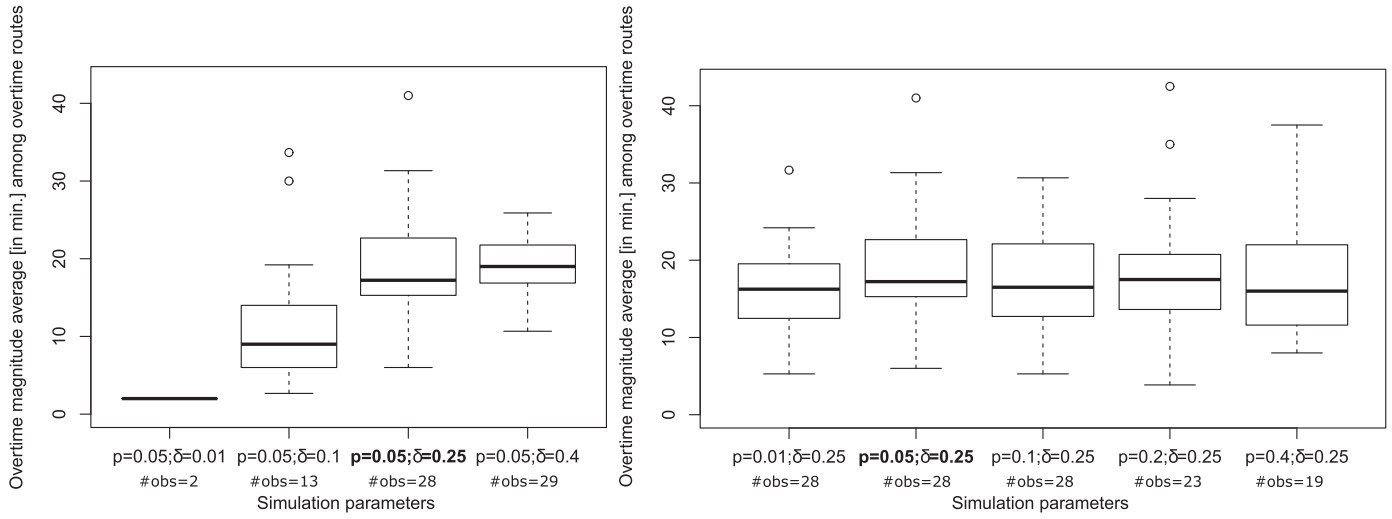


Fig. 9. Average overtime duration in minutes for different values of δ (left) and p (right).

To better apprehend the importance of the phenomenon, we next focus on the routes ρ_j in overtime. For this purpose, we introduce the subset $\mathcal{S}^{over}$ that encompasses all the scenarios in $\mathcal{S}$ with at least one route in overtime, that is $\mathcal{S}^{\text{over}} = \{s \in \mathcal{S} : R^{\text{over}}(s) > 0\}$. The magnitude of the overtime for ρ_j is the (positive) difference $A_{K_j+1} - \beta_j$. For each instance, KPI_7 is defined as the average, over all the scenarios in $\mathcal{S}^{\text{over}}$, of the average overtime among the routes in overtime:

$$KPI_7 = \frac{1}{\#\mathcal{S}^{\text{over}}} \sum_{\text{scenarios } s \in \mathcal{S}^{\text{over}}} \times \left(\frac{1}{R^{\text{over}}(s)} \sum_{\text{route } j \text{ in overtime in sol}(s)} \left[A_{K_j+1} - \beta_j \right] \right).$$

If an instance has no route in overtime for any scenario (i.e. $\mathcal{S}^{\text{over}}$ is empty), this instance is not considered in the boxplot in Fig. 9. The number of observations in each boxplot is indicated for each pair of parameters (p, δ).

One can see in Fig. 9 that the average overtime rarely exceeds 30 minutes, which appears to be an acceptable value in practice (especially, in view of the low frequency of such extensions). Here again, we observe that this overtime is partially "compensated" by a decrease of the route length for other drivers, but KPI_7 does not take this compensation into consideration.

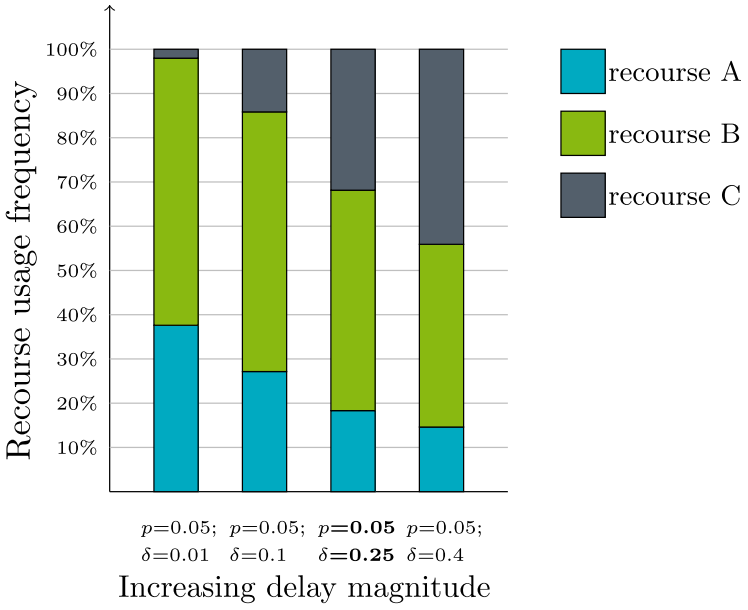
6.2. Algorithm analysis

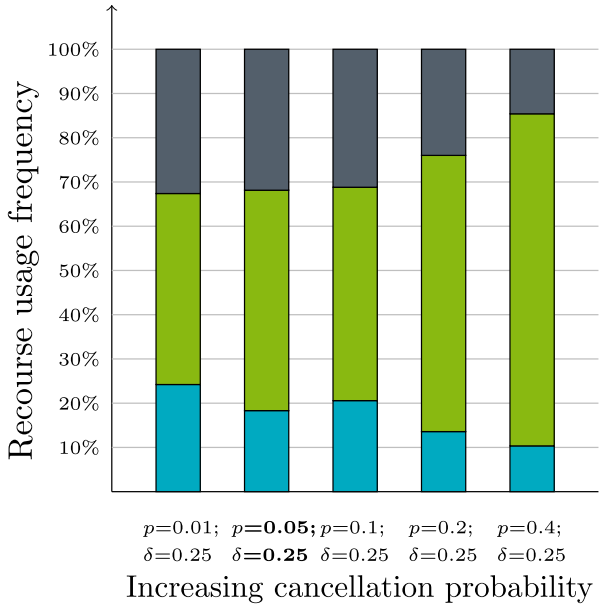
In this section, we focus on the behavior of the algorithmic approach itself, rather than on the quality of the solutions that it delivers. We first examine how the delays are handled in the scenarios and next, we discuss the effectiveness of each recourse operator.

6.2.1. Recourse utilization

As explained in Section 4.3, the increase of an appointment duration may lead to three different types of recourse: recourse A - no impact delay, recourse B - delay, recourse C - postponed action. (We do not explicitly mention recourse D here, since the number of applications of recourse D is always equal to the number of applications of recourse C, each postponed action giving rise to a subsequent reinsertion). We now analyze the usage frequency of each type of recourse as a function of the simulation parameters. More precisely, for a given pair (p, δ), we compute the frequency of cases where each type of recourse is used, as a proportion of all occurrences of increased appointment durations generated in all the scenarios for all the instances. The resulting statistics are displayed in Fig. 10.

On the left-hand side of Fig. 10, one can see that for small values of δ, recourse B is the most frequent one, and very few delays lead to a postponed action (and thus to a reinsertion attempt). When δ increases, the use of recourse C increases while





Use	p=0.05			
	δ=0.01	δ=0.1	δ=0.25	δ=0.4
Rec. A	37.61%	27.15%	18.32%	14.61%
Rec. B	60.37%	58.69%	49.82%	41.30%
Rec. C	2.02%	14.16%	31.86%	44.09%

Use	δ=0.25				
	p=0.01	p=0.05	p=0.1	p=0.2	p=0.4
Rec. A	24.22%	18.32%	20.58%	13.58%	10.35%
Rec. B	43.17%	49.82%	48.22%	62.43%	75.07%
Rec. C	32.61%	31.86%	31.19%	23.99%	14.59%

Fig. 10. Frequency of recourse usage for different values of δ (left) and p (right).

recourse B is less frequently used. Indeed, as expected, when the delays get larger, it is more likely that a delayed subrequest cannot be served as initially planned, but will require a reinsertion and thus a subsequent route modification. An interesting observation is that even when the delays are large, more than 55% (frequency of recourses A and B) of them can be handled without modifying the routes, and more than 10% (recourse A) do not even require a schedule adjustment. Regarding the right-hand side of Fig. 10, one can see that when p increases, the proportion of recourse B increases as well, while the other two recourse proportions decrease. The proportion of recourse A probably decreases because if there are more cancellations, then there is more time between consecutive subrequests, and thus the driver can arrive early at the next subrequest location, say l_k. This implies that H_k (the planned pickup time) tends to decrease, and hence, a delayed patient is more likely to terminate her appointment *after* H_k. Such an event typically gives rise to a recourse of type B rather than one of type A: the patient delays are more likely to be perceived even if they do not require adaptations of the remaining part of the routes. The proportion of recourse C also decreases when p increases because if there are fewer subrequests in the routes, the driver can wait longer for delayed patients. Finally, for large values of p, one can see that more than 80% of the delays do not require a modification of the routes.

6.2.2. Effectiveness of reinsertion operators

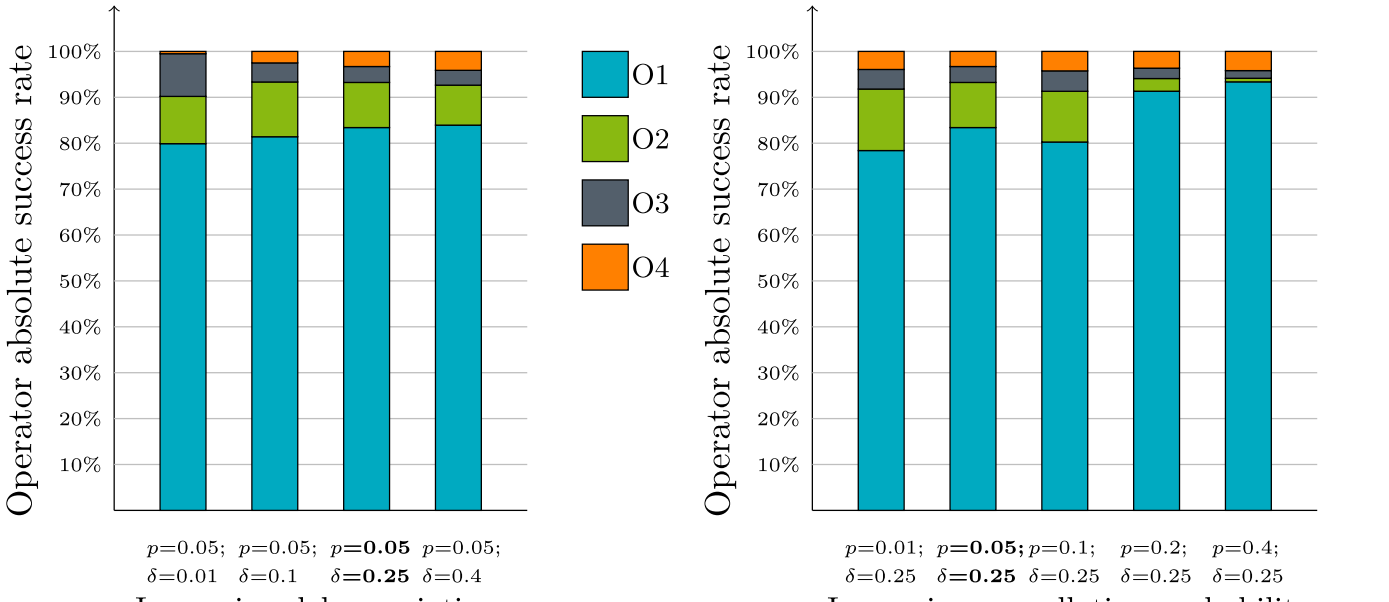
Let us now analyze the effectiveness of the individual operators in two different ways: first, we consider the contribution of each operator to successful reinsertions; next, we measure the success rate of the operator when an attempt is made to apply it.

Absolute success rate of operators. For each pair of parameters and for each reinsertion operator, Fig. 11 displays the proportion of successful reinsertions performed by the operator over the *total* number of *successful* reinsertions, that is, the absolute success rate of the operator. In other words, it represents the part of success due to each of the reinsertion operators. For example, among all successful reinsertions achieved when $(p, \delta) = (0.05, 0.25)$ (for 50 scenarios $\times 30$ instances), 83.41% have been performed with O1, 9.82% with O2, 3.47% with O3, and 3.29% with O4. One can observe that most of the time, the successful reinsertions are achieved with the simple reinsertion operator O1 (more than 78%). This first operator is especially effective when p is large (right-hand side of Fig. 11), while its performance is stable for varying values of δ (left-hand side of Fig. 11). Since the operators are called in hierarchical order, fewer reinsertions are attempted with the last operator O4 and moreover, only the most complicated reinsertions remain at that point. However, for large delays (that is, for large values of δ), more than 4% of feasible reinsertions are achieved with operator O4. As a conclusion of this analysis, all four operators prove useful and are able to identify feasible reinsertions in certain situations.

As a side-remark, one can notice the connections between the analysis of KPI_4 in Fig. 6 and the frequency of O3 in Fig. 11. The left-hand sides of these figures show that, when δ increases, the proportion of extended time windows increases (KPI_4), but in Fig. 11, the absolute success rate of O3 actually decreases. It means that even if there is a larger proportion of extended time windows when δ increases, O3 is less able to successfully reinsert the requests. This opposite trend is likely to be due to the fact that several time windows may have been extended in order to reinsert only one request. When p increases, the proportion of extended time windows decreases in Fig. 6, which is reflected in the decrease of the proportion of successful reinsertions brought by O3 in Fig. 11.

A similar connection can be established between KPI_6 in Fig. 8 and the frequency of O4 in Fig. 11. The increasing values of KPI_6 for increasing values of δ are indeed reflected in the left-hand side of Fig. 11, since a larger proportion of successful reinsertions are due to O4.

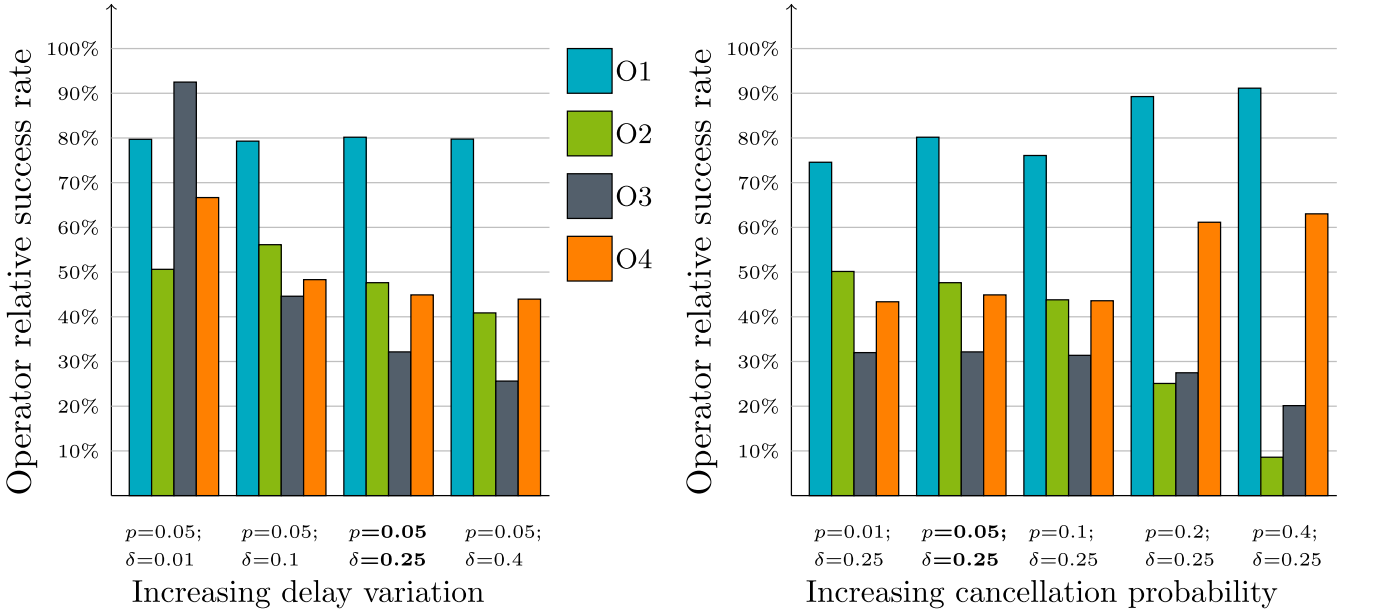
Relative success rate of operators. For each pair of parameters and for each reinsertion operator, Fig. 12 displays the proportion of



Success	$p=0.05$			
	$\delta=0.01$	$\delta=0.1$	$\delta=0.25$	$\delta=0.4$
O1	79.90%	81.41%	83.41%	83.94%
O2	10.30%	11.93%	9.82%	8.71%
O3	9.30%	4.16%	3.47%	3.23%
O4	0.50%	2.50%	3.29%	4.12%

Success	$\delta=0.25$				
	$p=0.01$	$p=0.05$	$p=0.1$	$p=0.2$	$p=0.4$
O1	78.40%	83.41%	80.26%	91.33%	93.37%
O2	13.41%	9.82%	11.05%	2.76%	0.78%
O3	4.26%	3.47%	4.45%	2.26%	1.67%
O4	3.93%	3.29%	4.24%	3.65%	4.18%

Fig. 11. Operator absolute success rate for different values of δ (left) and p (right).



Success	$p=0.05$			
	$\delta=0.01$	$\delta=0.1$	$\delta=0.25$	$\delta=0.4$
O1	79.70%	79.29%	80.17%	79.75%
O2	50.62%	56.13%	47.63%	40.87%
O3	92.50%	44.60%	32.15%	25.62%
O4	66.67%	48.31%	44.90%	43.95%

Success	$\delta=0.25$				
	$p=0.01$	$p=0.05$	$p=0.1$	$p=0.2$	$p=0.4$
O1	74.57%	80.17%	76.08%	89.26%	91.14%
O2	50.14%	47.63%	43.80%	25.09%	8.59%
O3	31.99%	32.15%	31.38%	27.48%	20.13%
O4	43.36%	44.90%	43.59%	61.15%	63.03%

Fig. 12. Operator relative success rate for different values of δ (left) and p (right).

successful reinsertions over the number of reinsertions *attempted* with this operator, i.e., the relative success rate of the operator over all the scenarios and all the instances. In other words, it represents how effectively each reinsertion operator performs when an attempt is made to apply it.

On the left-hand side of Fig. 12, one can see that the first operator succeeds in more than 79% of the attempted reinsertions with this operator. This number seems quite stable with respect to variations of δ. The destroy-and-repair operator O2 seems to be slightly less effective when the delays are large. However, as explained already, because of their hierarchical use, all the operators are not tested on the same instances: O2 is tested only on the scenarios that failed with O1, O3 only on the scenarios that failed with O2 (and thus, O1), and similarly for O4. Therefore, the last three operators are tested on fewer occasions (i.e., the denominator is smaller) and on these occasions, the reinsertions are likely to be quite difficult. The third operator O3 gives really high relative success rate for small value of δ but this rate decreases when the delays increase. The limitations imposed on the time window extensions (values of η and ζ in Table 4) are probably responsible for this behavior. The relative success rate of the fourth operator O4 tends to slightly decrease with δ as well. However, O4 is still able to perform 40% of the attempted reinsertions when $\delta = 0.4$. This observation is particularly important since O4 is the last operator applied before declaring a failure, i.e., before denying service to a patient.

The right-hand side of Fig. 12 shows that O1 is able to achieve a success rate of 91.14% when the cancellation probability is large. As a consequence, less reinsertions have to be attempted with the other three operators. One can observe that operators O2 and O3 have a decreasing relative success rate while O4 seems to be more efficient when p increases. This observation may be due to the features of the solutions. When p gets large, there are fewer subrequests in the routes and thus more time between consecutive subrequests (on average). This explains the high success rate of O1. For the reinsertions that O1 is not able to manage, the difficulty is not due so much to the tightness of the sequence of subrequests (that O2 and O3 directly address), but rather to the end of the driver working hours. This would explain why O4 is able to achieve 60% of the remaining reinsertions. Modifying the value of ξ in Section 4.3 may potentially lead to even higher success rates for O4.

In any case, a conclusion of this analysis is again that all four operators perform quite effectively when a reinsertion is attempted. This suggests that they are all relevant under different circumstances.

7. Conclusion

This article presents a reactive algorithm for recovery management in a dial-a-ride problem with real-time disruptions. Starting from an initial plan to serve the patient requests, a collection of procedures are proposed to manage the disruptions occurring along the day, namely, the patient delays and cancellations. In real-world settings, such an algorithm must quickly provide a solution in order to enable the dispatcher to react efficiently and to recover from the disruptions. Moreover, relatively minor disruptions created by a last-minute cancellation or by a delayed appointment should not have global repercussions on all the drivers and patients. For these reasons, our approach favors simple, local modifications of the current plans, rather than, say, a complete reoptimization of the plans whenever a disruption occurs.

The approach is accordingly built upon several types of recourses that can handle the disruptions when they are revealed. In particular, when a patient is too late for the driver to be on time for the following patients, then the request of the delayed patient is temporarily postponed until she is ready for her inbound journey. In the algorithm, this situation is handled as a reinsertion of the request. Reinserting a request can be difficult and therefore, four operators are developed to maximize the possibility of identifying successful reinsertions. When several reinsertions are available, then the selection is based on the slack time (so as to facilitate later reinsertions) or service quality factors, such as the minimization of the time windows.

The computational experiments, based on discrete event simulations, show that the algorithm is very fast (less than 2 seconds for a whole day simulation) and that the use of local reinsertion operations is most often sufficient, since the number of unassigned patients at the end of the day is extremely low. The frequency and severity of the patient time windows extensions and driver overtime are also analyzed and appear to be acceptable in practice. Moreover, the local operators bring only small changes in the solutions as required by the application. The results show the effectiveness of this approach in supporting the decision-making process for real-time replanning operations.

An additional benefit of the chosen approach is its flexibility. Indeed, several extensions or modifications could easily be implemented, such as different service durations for loading and unloading operations, service durations dependent on the locations (e.g., if the patient requires special assistance at home), or drivers' breaks.

In practice, route congestion may have an impact on the travel times. In our applied context, all trips take place in a local urban area where the travel times may depend on the time of day, but where huge traffic jams are unlikely to happen. Our algorithm could be easily adapted in order to deal with travel times that depend on the period of the day (peak hours, holidays, etc.). Accounting for random travel times, however, would require different extensions of our approach.

Acknowledgments

This research has been supported by the project PRESupply funded by the Walloon Region of Belgium (Convention #7566). The authors are grateful to the academic and industrial partners of the project for their helpful collaboration. In particular, they thank Charles Thomas for his help in generating the instances used in the computational experiments. The authors also acknowledge useful comments by an anonymous reviewer.

References

Berbeglia, G., Cordeau, J.-F., & Laporte, G. (2010). Dynamic pickup and delivery problems. *European Journal of Operational Research, 202*(1), 8–15.

Berbeglia, G., Pesant, G., & Rousseau, L.-M. (2011). Checking the feasibility of dial-a-ride instances using constraint programming. *Transportation Science, 45*(3), 399–412.

Berbeglia, G., Pesant, G., & Rousseau, L.-M. (2012). Feasibility of the pickup and delivery problem with fixed partial routes: A complexity analysis. *Transportation Science, 46*(3), 359–373.

Cappart, Q., Thomas, C., Schaus, P., & Rousseau, L.-M. (2018). A constraint programming approach for solving patient transportation problems. In J. Hooker (Ed.), *Principles and practice of constraint programming* (pp. 490–506). Cham: Springer International Publishing.

Cordeau, J.-F. (2006). A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research, 54*(3), 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*(6), 579–594.

Cordeau, J.-F., & Laporte, G. (2007). The dial-a-ride problem: Models and algorithms. *Annals of Operations Research, 153*(1), 29–46.

Coslovich, L., Pesenti, R., & Ukovich, W. (2006). A two-phase insertion technique of unexpected customers for a dynamic dial-a-ride problem. *European Journal of Operational Research, 175*(3), 1605–1615.

Häll, C. H., Högberg, M., & Lundgren, J. T. (2012). A modeling system for simulation of dial-a-ride services. *Public Transport, 4*(1), 17–37.

Ho, S. C., Szeto, W., Kuo, Y.-H., Leung, J. M., Petering, M., & Tou, T. W. (2018). A survey of dial-a-ride problems: Literature review and recent developments. *Transportation Research Part B: Methodological, 111*, 395–421.

Jaw, J.-J., Odoni, A. R., Psaraftis, H. N., & Wilson, N. H. (1986). A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Research Part B: Methodological, 20*(3), 243–257.

Liebchen, C., Lübbecke, M., Möhring, R., & Stiller, S. (2009). The concept of recoverable robustness, linear programming recovery, and railway applications. In R. K. Ahuja, R. Möhring, & C. D. Zaroliagis (Eds.), *Proceedings of the robust and online large-scale optimization, LNCS 5868* (pp. 1–27). Berlin Heidelberg: Springer-Verlag.

Liu, C., Aleman, D. M., & Beck, J. C. (2018). Modelling and solving the senior transportation problem. In W.-J. van Hoeve (Ed.), *Proceedings of the Integration of constraint programming, artificial intelligence, and operations research, CPAIOR 2018, Lecture Notes in Computer Science: 10848* (pp. 412–428). Cham: Springer.

Mitrović-Minić, S., Krishnamurti, R., & Laporte, G. (2004). Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological, 38*(8), 669–685.

Mitrović-Minić, S., & Laporte, G. (2004). Waiting strategies for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological, 38*(7), 635–655.

Molenbruch, Y., Braekers, K., & Caris, A. (2017). Typology and literature review for dial-a-ride problems. *Annals of Operations Research, 259*(1), 295–325.

Paquette, J., Cordeau, J.-F., & Laporte, G. (2009). Quality of service in dial-a-ride operations. *Computers & Industrial Engineering, 56*(4), 1721–1734.

Pillac, V., Gendreau, M., Guéret, C., & Medaglia, A. (2013). A review of dynamic vehicle routing problems. *European Journal of Operational Research, 225*(1), 1–11.

Psaraftis, H. N. (1986). Scheduling large-scale advance-request dial-a-ride systems. *American Journal of Mathematical and Management Sciences, 6*(3-4), 327–367.

Psaraftis, H. N., Wen, M., & Kontovas, C. A. (2016). Dynamic vehicle routing problems: Three decades and counting. *Networks, 67*(1), 3–31.

Ropke, S., Cordeau, J.-F., & Laporte, G. (2007). Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks, 49*(4), 258–272.

Savelsbergh, M. W. P. (1985). Local search in routing problems with time windows. *Annals of Operations research, 4*(1), 285–305.

Savelsbergh, M. W. P. (1992). The vehicle routing problem with time windows: Minimizing route duration. *ORSA Journal on Computing, 4*(2), 146–154.

Savelsbergh, M. W. P., & Sol, M. (1995). The general pickup and delivery problem. *Transportation Science, 29*(1), 17–29.

Schilde, M., Doerner, K. F., & Hartl, R. F. (2011). Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports. *Computers & Operations Research, 38*(12), 1719–1730.

Thomas, C., Cappart, Q., Schaus, P., & Rousseau, L.-M. (2018), CSPLib problem 082: Patient transportation problem. http://www.csplib.org/Problems/prob082.

Tukey, J. W. (1977). *Exploratory data analysis*. Reading, MA: Addison-Wesley.