CIRRELT

CIRRELT-2023-32

Healthcare Dynamic and Stochastic Transportation

Imadeddine Aziez
Mohamed Zied Torkhani
Jean-François Côté
Paolo Landa
Leandro C. Coelho

September 2023

Document de travail également publié par la Faculté des sciences de l'administration de l'Université Laval, sous le numéro FSA-2023-007

Bureau de Montréal
Université de Montréal
C.P. 6128, succ. Centre-Ville
Montréal (Québec) H3C 3J7
Tél : 1-514-343-7575
Télécopie : 1-514-343-7121

Bureau de Québec
Université Laval,
2325, rue de la Terrasse
Pavillon Palasis-Prince, local 2415
Québec (Québec) G1V 0A6
Tél : 1-418-656-2073
Télécopie : 1-418-656-2624

Healthcare Dynamic and Stochastic Transportation

Imadeddine Aziez[1,2], Mohamed Zied Torkhani[2], Jean-François Côté[1,2,*], Paolo Landa[1,2], Leandro C. Coelho[1,2]

1. Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)
2. Department of Operations and Decision Systems, Université Laval, Québec, Canada

Abstract. Transporting patients between different healthcare units in large hospitals is indispensable. Beds and wheelchairs are usually used to transport patients in one-building hospitals; however, campus-based ambulances and mini-vehicles are needed in many-building hospitals to perform patient transportation activities. In many cases, this service is managed manually, which increases the incurred costs, while there is room for improvement. This study investigates the problem of transporting people in a healthcare context, also known as the stretcher-bearer problem. The goal is to propose an approach that helps to build routes and assign incoming transportation requests to a hospital fleet of vehicles in real-time while minimizing the total weighted lateness of the delivery and the travel time of all vehicles. This study considers a multi-trip pickup and delivery problem with soft time windows, heterogeneous fleet, and capacity constraints. A static, a dynamic, and two stochastic approaches are proposed and tested using real-life data. The computational experiments demonstrated that a branch-and-regret approach outperforms the other approaches. A detailed sensitivity analysis is conducted to demonstrate the impact in practice of varying the number and capacity of the vehicles. Finally, the impact of having information a few minutes in advance is also demonstrated, which shows that the waiting and travel times can be significantly reduced.

Keywords: transportation of patients, dynamic pickup and delivery problem, reoptimization heuristic, scenario-based planification approach, branch-and-regret, waiting and relocation strategies.

Acknowledgements. This work was partly supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grants 2019-00094 and 2021-04037. We thank Compute Canada for providing high-performance parallel computing facilities.

---

* Corresponding author: jean-francois.cote@fsa.ulaval.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2023

1 Introduction

An efficient healthcare industry is one of the most vital sectors to nations' well-being and prosperity. Healthcare accessibility is one of the key dimensions of the human development index (HDI) [18]. Hence, worldwide expenses in the sector account for 9.58% of the global GDP as of 2018 [22]. The recent COVID-19 pandemic has not only emphasized the importance of efficient healthcare systems but, most importantly, has shown the precarity and weakness of this sector worldwide. Therefore, the need for better management of healthcare systems is imperative. According to Landry and Philippe [12], managing internal logistics in hospitals is very expensive and represents about 46% of the total management cost.

In large hospitals, patients are usually moved from one service to another, in the same or at a different building. In this type of large facility, poorly managed transportation activities may negatively impact the cost and the quality of the provided healthcare services. For example, a slight delay in transporting a patient for surgery can affect the entire schedule of the operating theater and the schedule of the medical staff assigned to perform the surgery. Transportation operations in hospitals are usually managed by a dedicated department, which assigns the received transportation requests from different departments of the hospital to the available fleet of vehicles. Urgent requests have priority over other requests. Each time a request arrives, it is immediately assigned to an available vehicle while considering the required vehicle type. Each department inside the hospital has a specific goal: minimizing the total lateness, travel time, distance, or transportation cost. The demand is unpredictable and is revealed dynamically during the day. New requests are integrated into the ongoing routes as they arrive. Building routes depends on the received information, such as patients' delivery time window, available medical staff, the number of required seats as well as the level of urgency.

The case study presented in this paper is based on a large hospital in Bologna, Italy. This hospital is composed of 30 buildings, each providing a different service. Each year, this facility receives thousands of patients requiring different treatments. All these flows lead to approximately 425 daily requests for transportation. To satisfy this demand, the hospital has a transportation service that manages five vehicles (three electric mini-vehicles and two ambulances). Around 90% of the demand is unknown at the beginning of the shift and revealed dynamically throughout the day. The revealed information at the arrival of each request includes the number of required seats and the delivery time window to be respected.

In this paper, we tackle several of these issues, namely determining the type and which vehicle to use, the vehicles' routes, and the vehicles' relocation strategy after serving their requests. We have access to our partner's historical data and propose algorithms to make these decisions in real time. This is a highly dynamic problem in which new information is revealed gradually during the day. We take that into account after first proposing a deterministic algorithm with full knowledge (oracle) that serves as a benchmark. We then propose relocation strategies for the vehicles to try to take advantage of preemptively placing them at different locations. We then develop dynamic and stochastic optimization algorithms to solve this complex problem. Using real data from the Bologna hospital, we tackle the dynamic problem of transporting patients where the goal is to minimize the total weighted lateness of the delivery and travel time of the vehicles. Hence, our contributions consist of the following:

- introducing a dynamic reoptimization algorithm adapted for our case study that uses a regret heuristic, a local search heuristic, and an adaptive large neighborhood search (ALNS);
- introducing waiting and relocation strategies adapted for this problem;
- developing a scenario-based planning approach (SBPA) that considers potential requests sampled by a probability distribution and a consensus function that uses stochastic information to relocate vehicles after considering all scenarios;
- proposing a branch-and-regret (B&R) heuristic that evaluates a set of relocation moves to select the best relocation move for each vehicle.

The remainder of the paper is organized as follows. Section 2 presents the literature review of pickup and delivery problems and the main works related to patients' transportation problems. Section 3 provides an overview of the case study. Section 4 provides a formal description of the problem. Section 5 is devoted to the solution approach. The results are presented in Section 6, and conclusions follow in Section 7.

2 Literature review

Problems considering the transportation of people are mainly related to the dial-a-ride problem (DARP), a particular case of the pickup and delivery problem (PDP). Unlike other PDPs, the

DARP considers both the total transportation cost and the users' satisfaction by minimizing or limiting the ride time for each passenger [8, 19, 20], which is particularly relevant in the healthcare sector.

Melachrinoudis et al. [13] proposed a DARP with soft time windows applied to a health center in the United States. The objective is to minimize the total transportation costs and clients' inconvenience. This problem was modeled using mixed-integer programming and solved using a tabu search metaheuristic. The problem was studied in its static and deterministic version, assuming all transportation demands are known. Hanne et al. [7] studied a dynamic DARP with specific hospital constraints such as patient inconvenience constraints (e.g, maximum ride time) and hospital service constraints (e.g., single transportation for isolation patients). They designed a scheduling system that supports all phases of the transport flow within hospitals, including transportation request booking, scheduling, dispatching, monitoring, and tracking trips in real time. To evaluate the quality and the cost of the solutions, the authors defined four performance criteria: total lateness, total earliness, total driving time, and total transport time for patients. They proposed several optimization algorithms requiring different solution times. This planning system has improved patients' general satisfaction and reduced transportation costs by 20%.

Beaudry et al. [2] also solved the problem of dynamic transportation of patients. In contrast to a basic dynamic DARP, the problem encompasses additional hospital-specific features, including order priorities, vehicles with alternative loading modes and desired rest periods, assistance of medical personnel and need for special equipment during transportation, and specific requirements for patient isolation to prevent the spread of infections. The main goal is to provide an efficient and timely transportation service between different departments inside the hospital. It is a multi-objective problem, aiming at ensuring the punctuality of the service by minimizing the total weighted travel time, the total lateness, and the total earliness. The authors proposed a two-phase heuristic that generates a feasible solution in the first phase using an insertion heuristic. This solution is then improved during the second phase using a tabu search. The results from real data provided by a large hospital in Germany show that this algorithm can handle the dynamic aspect of the problem, yielding high-quality solutions. It reduces the total patient waiting time using a minimum number of vehicles. Kergosien et al. [11] presented a dynamic PDP for transporting patients between care units in a hospital complex in Tours, France. Each request requires a specific type of vehicle, while the capacity is limited to one person at a time.

Subcontracting transport operations is possible but comes at a high cost. Building routes for the patients must also respect specific constraints and procedures of the hospital. The hospital prioritizes urgent requests and requires vehicle disinfection after transporting a patient with a contagious disease. The objective is to minimize transportation costs for the hospital and private companies. For this purpose, the authors developed a tabu search algorithm. Experiments with real and randomly generated instances have shown that this approach can provide high-quality solutions for this dynamic problem with an average computational time of less than 5 seconds. To the best of our knowledge, this is the only work that addressed the problem of patient transportation as a dynamic PDP, and this work inspires our problem modeling.

In the operations research literature, stochastic and dynamic vehicle routing problems (SDVRPs) have been studied since the 1980s. However, the interest of researchers in these problems has only increased recently as the availability of data and computing power increased [17]. In early research on dynamic vehicle routing problems, reoptimization methods were the basis for decision-making. These methods do not consider the stochastic aspect and instead use a rolling horizon to choose a route plan for the static routing problem at each period that contains known information. The authors explain that in these problems, identifying feasible routes can be challenging, thus, anticipating future uncertainties often becomes secondary to managing known information. Among these works, we mention Gendreau et al. [6], who studied the dynamic vehicle routing problem with soft time windows inspired by a courier service case, and Aziez et al. [1], who studied the fleet sizing and routing problem with synchronization for automated guided vehicles within a hospital with dynamic demands.

With the development of technology, particularly the increase in performance and computing power, the use of lookahead algorithms has increased. These are dynamic algorithms that consider the problems' stochastic aspects. Typically, they use potential scenarios to create route plans. One of the well-known lookahead algorithms is that of Hvattum et al. [9], who considered a VRP where both the location and the customer's demand can be unknown. This problem was modeled as an SDVRP and was solved using a dynamic stochastic hedging heuristic (DSHH) that employs sampled scenarios. This method samples scenarios for each time interval and solves them to construct a plan for the vehicles. The comparison with a reoptimization heuristic shows that the DSHH successfully reduced the total traveled distance by up to 15%, though it required an increase in the number of vehicles to achieve these results. To overcome this

drawback, Hvattum et al. [10] presented a B&R heuristic for the SDVRP, which was motivated by a real problem of a major transporter in Norway. The new approach was compared with the reoptimization heuristic and provided better results, but a greater traveled distance than the DSHH. An additional comparison was conducted with the multiple scenario approach (MSA), which is based on continuously generating routing plans for scenarios including known and stochastic customer demands. The results show that the MSA has better results than B&R regarding the number of unserved customers.

Consensus functions have also been used to study SDVRPs, including Bent and Van Hentenryck [4], Voccia et al. [21], Song et al. [16], and Côté et al. [5] to assign a score for each solution based on several features present in the solutions of the other scenarios. The function receives as an input the set of routing plans generated for all scenarios and returns a score for each one of them. The goal is to choose the solution with the highest probability of being implemented at the lowest cost in all scenarios. Voccia et al. [21] used a consensus function that counts the number of times the routes of a plan appear in other plans, whereas Côté et al. [5] used two consensus functions; the first one is called the Assignments Similarity, that for each request-route pair of a plan, computes a score that represents the number of times the pair appears in other plans. The plan with the highest score is then selected. The second function, called Edit Distance, computes a score for each plan that represents the sum of the number of changes needed to get each other. This second consensus function selects the plan with the minimum score.

Additional techniques to manage dynamism in VRPs are waiting and relocation strategies. A waiting strategy makes a vehicle stay at its current location for a certain time instead of moving to its next known customer. In contrast, a relocation one moves vehicles to a location where demand is expected to appear. The decision to wait or to relocate anticipates potential future demands to reduce delays. Mitrović-Minić and Laporte [14] developed and compared four waiting strategies to deal with a dynamic PDP arising from courier companies. These waiting strategies are based on two types of decisions, either “drive first” or “wait first”. Bent and Van Hentenryck [3] solved an online SDVRP where the objective is to maximize the total number of served customers. To achieve this, the authors developed an algorithm that uses waiting and relocation strategies. The algorithm solves sampled scenarios, and based on the best evaluation found, it decides whether or not a vehicle should wait and where it should

relocate. Results have proven the effectiveness of these strategies and have shown that their implementation improves the number of served customers.

Over the past years, the problem of patients' transportation in hospitals has attracted more attention from researchers. This growing interest can be explained by several factors, such as information technology development, patient satisfaction prioritization, and the need to reduce the increasing costs of internal logistics in hospitals. Different approaches have been proposed in the literature to solve this problem, aiming to increase patient satisfaction first, then decrease transportation costs. These approaches vary between linear programming, simulation models, static, and dynamic and stochastic algorithms. In this paper, motivated by the case of the Bologna hospital, we study the dynamic transportation of patients. This setting is complex and characterized by highly uncertain and dynamic demands. We propose four approaches to tackle the problem: the first one solves the static problem (oracle), the second is a myopic approach dedicated to solving the dynamic problem, and the last two are dynamic and stochastic approaches that take advantage of waiting and relocation strategies. Although these algorithms are developed to solve a specific case study, they can be applied to similar problems.

3 Case study

The dynamic PDP considered in this study is inspired by the case of the hospital Sant'Orsola-Malpighi in Bologna, Italy. This hospital is composed of seven departments and 91 operative units. It is the largest healthcare facility in Italy, with a capacity of more than 1,758 beds and about 5,355 employees, 881 of which are physicians. The hospital receives approximately 72,000 inpatient cases annually and about 4 million outpatient visits. In terms of area, the hospital is a campus that occupies 21 hectares and receives every day about 20,000 people, including patients and staff. It extends over 1.8 km with 30 buildings around a central tree-lined avenue, as shown in Figure 1.

Due to its large area, the hospital offers a transportation service for the patients to receive healthcare services between its buildings. The service is provided by five vehicles: three electric vehicles and two ambulances. The transportation service is managed by the department of transportation, which receives transportation requests from all care units and dispatches drivers to the available vehicles. Each request received by the department contains information about

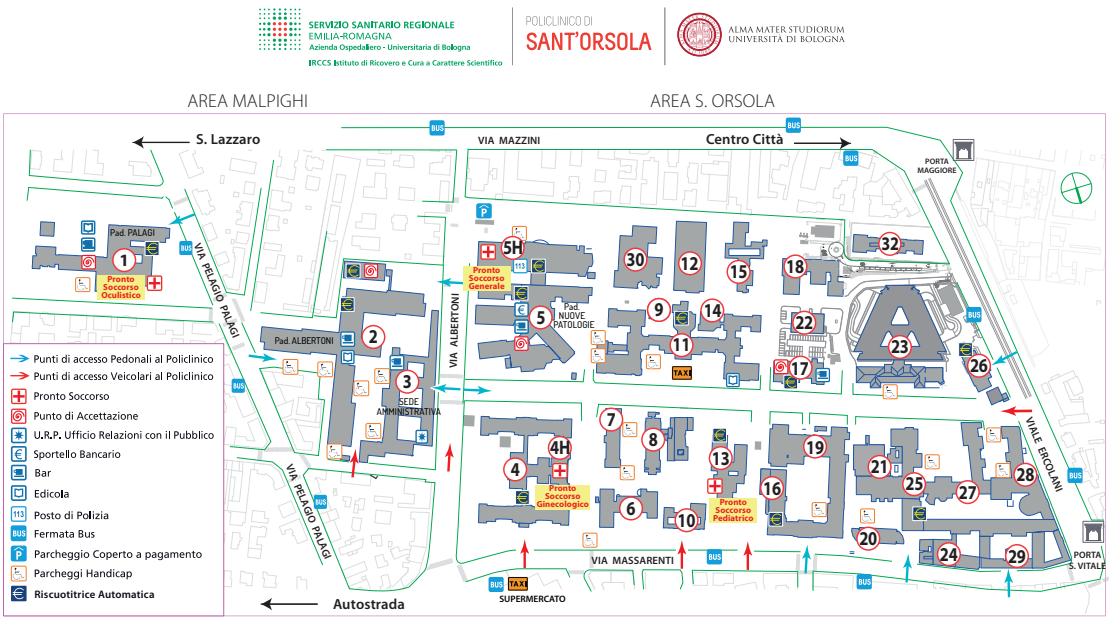


Figure 1: Plant of the Policlinico Sant'Orsola-Malpighi.

the patient, the start time of the medical service, the buildings and the floors where the patient must be picked up and then dropped off, the number of seats required, and the priority of the request based on the urgency of the case to be transported. If the request is known, the transportation service can schedule it in advance. Otherwise, if the request arrives suddenly during the day, it is scheduled in real-time. The department of transportation receives requests and operates 24 hours a day, but most requests come during the day.

For this study, we assume that a typical workday at the hospital starts from 7:00 A.M. to 8:00 P.M. Since the department receives a very small number of requests during the night, it can be easily managed. Therefore, requests received between 8:00 P.M. and 7:00 A.M. are disregarded. On the other hand, requests that have a release time before 7:00 A.M. and have a service start time after 7:00 A.M. are included in our planning as part of the requests known in advance. The requests that arrive during the day are unknown and give the dynamic aspect to our problem. Based on historical data collected from the hospital, we find that about 7.5% of the requests are known before starting the working day, while the rest, 92.5%, are unknown at the start of the shift and need to be planned in real-time as they arrive.

4 Problem description

Let $P = \{1, 2, ..., p\}$ be the set of p buildings of the hospital. Each building $i \in P$ is composed of f_i floors represented by $F(i) = \{1, 2, ..., f_i\}$. The problem can be represented by a complete graph $G = (N, A)$ where N is the set of nodes of all possible building-floor combinations, A is the set of all arcs in the graph. A driver transporting a patient between two points $i, j \in N$ travels through arc $(i, j) \in A$ where $i \neq j$.

Let $R = \{r_1, ..., r_n\}$ be the set of n transportation requests. For each request $r \in R$, we define a pair of nodes (i, j) where $i, j \in N$ are the patient's pickup and drop-off locations. Each pickup or drop-off node is associated with a unique request, allowing multiple nodes at the same location. Each request $r \in R$ is received at time t_r, also known as the release time, and has a soft time window $[a_r, b_r]$ where a_r specifies the time when the patient is available for the service (early time window), and b_r is the last instant during which the service can be done without lateness. Note that for all requests in R, we assume that $t_r \leq a_r$, and the default value is $t_r = a_r$. The width of the TW interval of each request depends on the urgency level of the service denoted by u_r. Thus, urgent requests have tighter TW intervals than less urgent requests. The assumed TW intervals and their corresponding levels of urgency are presented in Table 1. Finally, each request r is characterized by the number of seats w_r required to transport the patient, which depends on the health conditions of the patient.

Table 1: Levels of urgency

TW interval (min)	u_r
5	5
20	4
30	3
60	2
> 60	1

A fleet of heterogeneous vehicles $K = \{1, ..., k\}$ is available to transport the patients. It is composed of two types of vehicles: ambulances and electric vehicles. Depending on the type denoted by q, each vehicle has a capacity c_q. Ambulances are limited to one patient, so they are limited to trips between a single pickup location and a single drop-off location. However, electric vehicles can load up to four patients at a time. This gives them the flexibility to make multiple trips between multiple pickup and drop-off locations. The fleet is available from 7 A.M. to 8

P.M., and the same department also monitors driver rotation and changes. Therefore, there are no delays during the shift. Each vehicle starts and ends its working shift from a central depot $0 \in N$ located on the ground floor of the hospital's emergency ward. Vehicles have the option to wait or to be relocated during their shift.

Each arc $(i, j) \in A$ requires a travel time t_{ij}^{q}, where $q = \{1, 2\}$ depends on the vehicle type q and includes all the time required to get into the vehicle and engage into the route such as walking time, driving time, the time required for taking the elevator, and pickup and drop-off times. Furthermore, the paths taken by the vehicles are different since ambulances can only travel using external roads. In contrast, electric vehicles can use underground roads and external roads connecting buildings.

This problem aims to determine a set of routes to fulfill all the transportation requests while minimizing the total weighted lateness and travel times of all vehicles. The lateness of request r, denoted by l_r, is defined as the difference between τ_r (the time at which the delivery of the request r is completed) and b_r (the upper bound of the time window). The lateness l_r is only considered when its value is positive, i.e., $l_r = max\{0, \tau_r - b_r\}$. Given that fact that it is highly critical to delay urgent requests from a healthcare perspective, in the objective function, the value l_r of each request is multiplied by $\beta_r = \beta \times u_r$ which depends on the urgency level u_r of each request r, and the constant β is used to give the lateness a much greater importance than the travel time in the objective function, because the potential delays do not only cause a decrease in the patients' satisfaction but also have an impact on the hospital staff's schedule, especially those in direct contact with the patients. As indicated earlier, in healthcare transportation, patient satisfaction is more important than cost reduction.

A solution to the problem minimizes the total weighted lateness of the deliveries and the travel time of all vehicles, and ensures that all requests are satisfied, all routes start and end at the depot, vehicles capacities are respected, besides logical constraints such as respecting release times, opening time windows, and precedence constraints. It is important to highlight that this problem cannot be classified as a proper DARP for the missing maximum ride time constraint. It is motivated by the fact that time windows are very short and patients are onboard the vehicle for a short time, as the distances are also relatively short.

5 Solution methods

This section presents our proposed solution approaches to tackle the problem. We start by describing how the static variant of the problem is solved in Section 5.1, then a myopic approach for the dynamic problem is presented in Section 5.2; the myopic approach works on a rolling horizon by solving a series of static problems where only the information available at that time is used. Finally, we present the details of the dynamic and stochastic approaches in Section 5.3.

5.1 Static problem (oracle)

The static version of the problem assumes that all the requests are known in advance. This version of the problem represents the case where we have the best possible conditions, without uncertainty and with full knowledge of the data. An ALNS heuristic based on the one of Ropke and Pisinger [15] is developed to solve this static variant. An overview of the heuristic is given in Algorithm 1. It starts by creating an initial solution using an insertion heuristic. Then, removal-insertion operations are performed for η iterations. At each iteration, a random number q of requests to remove and a removal and an insertion operator are selected. Solution s is copied into s', and q requests are removed. The removed requests are then reinserted into the solution using the insertion operator. The best solution s^* is updated if s' is better than the current best solution. Otherwise, the incumbent solution s is updated if the acceptance criterion is met. The weights of the operators are then adjusted.

Algorithm 1: ALNS heuristic.

```
Construct an initial solution s
s* = s and initialize weights
for i = 1 to η do
    Generate a random number q of requests to remove
    Randomly select a removal and an insertion operator
    s' = s
    Remove q from s' using the removal operator
    Insert the unrouted requests in s' using the insertion operator
    if s' is better than s* then
        s* = s = s'
    else if s' satisfies an acceptance criterion then
        s = s'
    Adjust weights
return s*
```

The main characteristic of ALNS is that it explores several insertion and removal operators selected according to their past performance at improving solutions. The adaptive feature is possible through the weight associated with each operator. The weight of an operator is modified if it improves the solution. At each iteration, the ALNS chooses two operators using a roulette wheel selection based on those weights. In this paper, classical operators from Ropke and Pisinger [15] are used: *sequential* and *regret* insertion operators, as well as *random* and *related* removal operators. It is important to highlight that it is always possible to achieve a feasible solution due to soft time windows. The results of the static approach are the best possible results representing a best-case scenario. They also represent a lower bound for the lateness and travel time. However, this approach does not reflect the reality since it assumes that all the requests are known at the beginning of the day but serves as a benchmark. Particularly, when all the requests are known in advance, vehicles can preemptively travel toward the next pickup and wait for the time window to open. This relocation does not happen in real life and is discussed next.

5.2 Dynamic problem

This section describes the event management actions and the type of events that trigger the reoptimization process. We then give the details of the waiting and relocation strategies. Finally, we present the reoptimization heuristic designed to tackle the dynamic problem without considering any knowledge about future events.

5.2.1 Events management

As time unfolds, pickup and delivery operations are performed, and new requests become known. In our framework, each time a new event occurs, we allow different types of decisions to be taken:

- Routing decisions: these are the classical decisions of assigning requests to vehicles and building routes. These decisions must contain all the information of the assignments, such as the vehicle, the pickup and the delivery locations, and the information related to the patients (urgency, number of required seats).
- Strategic decisions: we allow two preemptive actions that do not involve any transportation requests. These decisions concern waiting and relocation strategies. They dictate

whether a driver should wait at their current location or be relocated elsewhere until a new assignment is received.

In the dynamic problem, the algorithm collects all the known information when one of the following events occurs:

- a new request is revealed and at least one vehicle is idle;
- a driver has completed a drop-off;
- a vehicle has completed a relocation.

A reoptimization step is performed each time one of these events occurs. We continuously maintain a solution, and as events occur, it is updated according to the actions carried out and the new data released. The parts of the solution that are already performed become permanently fixed. This includes pickups and deliveries and ongoing operations like waiting and moving to a new location. Then, a reoptimization is executed to determine the new decisions for all outstanding requests known at that time.

5.2.2 Waiting and relocation strategies

Whenever a new event occurs, the algorithm generates all known requests up to that point in time. Then, using the past decisions, the algorithm reconstructs the previous routes and performs a state check for each driver. Based on the state of the driver at that time, the algorithm makes a decision. If a driver has completed a delivery and has no further assignments, they can wait or be relocated. The relocation tries to anticipate potential future moves to optimize the objective function. If a driver is not assigned to any request, moving closer to a potential source of the next request is more advantageous, if one can anticipate such an event.

Based on the analysis of the real data provided by the hospital, we developed the following waiting and relocation strategies:

- **Strategy 1 – Wait:** this a waiting strategy where each driver with no assignment stays where they are until they are assigned to a new request, i.e., they remain on the floor of their last delivery.

- **Strategy 2 – Depot:** this strategy aims at relocating each vehicle to the depot in the emergency building where it will wait to be assigned to a new request.
- **Strategy 3 – Ground floor:** this is similar to the second strategy, but instead of relocating the vehicles to the depot, each driver moves back to the ground floor of their last delivery building.

In the first strategy, there are no additional decisions to include in the solution, and ALNS optimizes the routes while considering the last delivery positions. However, it may take long for the driver to move to the next pickup location, which can be explained by the time required to leave the floor or the building of its current location to travel to the next destination. The second strategy is currently used by the hospital in our case study. This strategy is adopted because most transport requests originate from the emergency building. However, in terms of the fleet distribution among the different buildings, it reduces the coverage and focuses on a limited campus area. In contrast to the second strategy, the third strategy relocates the drivers to their vehicles on the first floor of the last delivery building. It is aimed at increasing the coverage of vehicles all over the campus so that they remain closer to the various sources of requests.

We propose three different types of approaches to reoptimize the problem. The first approach uses *reoptimization* heuristics that do not use any forecasts on future requests. It solves a static problem where only known information is used at each event. This wait-and-see approach is described in Section 5.2.3; we then exploit stochastic information in Section 5.3. The second approach is based on SBPA heuristics inspired by Bent and Van Hentenryck [4] (Section 5.3.1). The third one is based on B&R heuristics inspired by Hvattum et al. [10] (Section 5.3.2).

5.2.3 Reoptimization heuristic

The dynamic approach is based on a reoptimization heuristic (RH) that does not consider the demand forecasts, i.e., the stochastic aspect. Although such a heuristic is theoretically simple, its use requires some modifications to properly integrate the waiting and relocation strategies presented in the previous section. This gave us three different versions of this heuristic denoted by: RH-W, which uses the waiting strategy; RH-D, which relocates the driver to the depot; and RH-GF, in which the driver waits at the ground floor of its last drop-off building.

The RHs are simple heuristics that do not consider any knowledge about future events. When an event occurs, past actions are fixed, and the solution is reoptimized using the ALNS algorithm. The variants RH-W, RH-D, and RH-GF are developed to test different vehicle waiting and relocation strategies. For these heuristics, the only event we consider is the arrival of a new request.

An overview of the RH is found in Algorithm 2. The algorithm starts by creating a solution containing the known requests at the beginning of the day. Then, the ALNS algorithm is executed to find an initial solution. Vehicles depart only if they can arrive at least at the start time of their first request, otherwise, they wait at their current location. Then, each time a new request is revealed, the completed and ongoing assignments are locked in the solution, meaning the completed pickups and deliveries are permanently locked in their routes. Only new and non-performed requests can be inserted after the fixed nodes or in empty routes. Each new request is integrated into solution s and is reoptimized. Vehicles without assignments are managed according to one of the waiting and relocation strategies. The solution s is then executed until a new request arrives.

Algorithm 2: Reoptimization heuristic.

```
Input: RH-W, RH-D, or RH-GF
Create a solution s that contains the known requests at t = 0
Optimize s with the ALNS algorithm
while there are new requests do
    Lock past assignments in s
    Add the new request to s
    Reoptimize s with the ALNS
    Apply the selected RH strategy to any waiting vehicle
```

5.3 Stochastic problem

This section presents the solution approaches designed to tackle the stochastic problem. The first is the scenarios-based planning approach, and the second is the branch-and-regret heuristic.

5.3.1 Scenarios-based planning approach

In the SBPA, potential scenarios that may occur in the near future are generated each time an event occurs. The scenarios are obtained by sampling performed past requests from a history

data or by sampling the probability distribution of their appearance. An optimization algorithm is executed to solve each scenario, creating routing plans. After that, a consensus function is used to select a plan. The Algorithm 3 describes the pseudo-code of the sampled scenario-based planning heuristic.

The first steps (lines 1–7) are similar to the ones in the RH algorithm. In line 8, the algorithm calls the function "FreeDriver" which returns true or false value. This function takes the current time t of the algorithm as a parameter and checks if there are any available drivers in the near future $(t + \Delta)$. It returns true if the function finds at least one available driver in the time bucket Δ. As a result, the algorithm generates a set of ω scenarios of potential requests that can occur between the time t and $t + \Delta$. For each scenario ω, the heuristic generates a plan s_ω that contains all the potential requests of the ω scenario. The plan s_ω is then optimized using ALNS. After that, each vehicle is relocated to the node with the highest score using the consensus function. If the "FreeDriver" function finds no available drivers in the near future, the SH strategy selected in line 1 is then applied. The SH strategies are those used in the RH: SH-W indicates the algorithm that uses the waiting strategy, SH-D indicates the algorithm that relocates the driver to the depot, and SH-GF indicates the algorithm where the driver waits at the ground floor of its last drop-off building.

The consensus function takes as an input the set of all plans created for all scenarios. In return, the function computes a score for each pair of vehicle-node and relocates each vehicle to the node with the highest score. The score is computed by counting for each vehicle the number of times each node appears in all routing plans. The goal is to relocate each vehicle to the location with the highest probability of being its next destination. Thus, this relocation method can be considered a third relocation strategy.

5.3.2 Branch-and-Regret Heuristic

The B&R heuristic was introduced by Hvattum et al. [10] to solve the SDVRP. It is a look-ahead algorithm that uses sample scenarios of potential future requests to better plan the actions taken by the vehicles. In the B&R, a set of scenarios Ω is generated at each new event, and plans are obtained by solving the scenarios. Each scenario is a possible realization of future events. In our case study, each scenario $\omega \in \Omega$ comprises known and potential future requests. The future requests are generated from historical data or a probabilistic distribution. The B&R maintains

Algorithm 3: Scenario-based planning heuristic.

```
Input: SH-W, SH-D, SH-GF, time bucket Δ
Create a solution s that contains the known requests at t = 0
Optimise s with the ALNS
while there are new requests do
    Lock past assignments in s
    Add the new request to s
    Reoptimise s with the ALNS
    if FreeDriver(s,t,Δ) then
        Generate a set of scenarios Ω of potential requests
        foreach scenario ω in Ω do
            Construct a solution s_ω which contains all the requests of the scenario ω
            Optimize s_ω with ALNS
        foreach Available vehicle k do
            Relocate k to the node with the highest consensus function score
    else
        Apply the selected SH strategy
```

a solution s that is executed until a new event occurs. Whenever a new event occurs, the set Ω of scenarios and a solution s_ω associated with each scenario are created. Then, s is updated using the knowledge collected from optimizing each solution s_ω.

The B&R heuristic starts by individually producing solutions to each sample scenario ω. The resulting solutions based on the set Ω will most likely differ because they contain different future requests. The heuristic then iteratively fixes some decisions in the sample scenario solutions until they all share a common structure. Fixing decisions is closely related to implementability (or non-anticipativity) in stochastic programming. The sample scenario solutions anticipate their own future and are not particularly good under different outcomes. A solution having non-anticipativity means it does not act on a specific future, and such a solution should behave well under different outcomes. In the B&R heuristic, non-anticipativity is added progressively, similarly to the partial exploration of a branch-and-bound tree. At each node of the tree, it creates and evaluates several branches. In our B&R heuristic, the branching consists of assigning an action to a vehicle, which can be wait or relocate. To evaluate a node, each solution s_ω is optimized, knowing that the assignment of a new action to a given vehicle is fixed in the solution. The node having the lowest average cost is selected and visited. When the solutions are sufficiently non-anticipative, the decisions taken are imposed to s and executed until a new event occurs.

Our work differs from the one of Hvattum et al. [10] in several ways. First, events are not accumulated and solved at each predetermined moment. Instead, we reoptimize the system at each event to avoid having vehicles, patients, and the staff waiting uselessly. Second, the decisions that are evaluated are also different. Hvattum et al. [10] branch on the vehicle's arrival time of the real requests. It starts by optimizing each scenario; they select a request served at different times in the sample scenario solutions. Their branching scheme consists of creating a branch where the request is served before a specific moment and another one where it is served after that moment. Both branches are evaluated by reoptimizing all scenarios, and the one having the lowest average cost is explored. They keep branching until each real request is either served before or after the specific moment. This saves computation time because only one path from the root to a leaf is evaluated. Once done, another round of branching occurs to make request-to-vehicle assignments.

The proposed B&R heuristic is described in Algorithm 4. Initially, a solution s containing all the known requests is created and optimized using ALNS. At each event, the past assignments are locked, and the new requests are added to s; after that, the plan is optimized again using ALNS. The next step is to check for free drivers; if there is at least one free driver, a set of scenarios Ω is generated. Each scenario ω comprises real and potential requests that may arrive later. An initial solution s_ω is created for each scenario and optimized using ALNS. Note that the visited nodes and those having a vehicle driving towards them are locked in their corresponding routes.

In the next step, we use the generated solutions to calculate the number of times each free driver is first assigned to perform a pickup in all scenarios. For simplicity, we refer to it as first pickup. Then, the free drivers are sorted in descending order based on the number of first pickups in the sample scenario solutions. After that, for each free driver v, we create the set D of all possible relocate decisions, which contains the locations of every first pickup, the location of the emergency's first floor, the ground floor of the driver's current location, and the driver's current location (waiting strategy). The next step is to decide what a free driver does next, representing the branching part in the B&R algorithm. For each decision $d \in D$ and each scenario $\omega \in \Omega$, the decision d is fixed in s_ω, which is then optimized using ALNS. The cost of each solution s_ω is used to calculate an average cost of assigning the driver v to perform the decision d. In the final step, the decision incurring the lowest average cost is implemented in s. We keep branching until all free drivers have an assignment. Our tree exploration scheme is heavier regarding computational

time than Hvattum et al. [10]. However, we can explore a broader range of decisions, and we may find better solutions. Finally, the solution s is reoptimized using ALNS.

Algorithm 4: Branch-and-Regret heuristic.

```
Input: time bucket Δ
Create a solution s that contains the known requests at t = 0
Optimise s with the ALNS
while there are new requests do
    Lock past assignments in s
    Add the new request to s
    Reoptimise s with the ALNS
    if FreeDriver(s,t,Δ) then
        Generate a set of scenarios Ω of potential requests
        foreach scenario ω in Ω do
            Set s_ω = s ∪ ω
            Optimize s_ω with ALNS
        Sort free drivers based on the number of pickups in all scenarios
        foreach free driver v do
            D ← Get possible relocate decisions of v
            foreach decision d in D do
                foreach scenario ω ∈ Ω do
                    s_ω = s_ω ∪ d
                    Optimize s_ω with ALNS
            Implement the least-cost decision d
    Optimize s with ALNS
```

6 Experimental results

This section presents the test instances and discusses the results of our solution approaches. Section 6.1 explains the generation of scenarios. Section 6.2 is dedicated to parameter setting. Section 6.3 presents the results obtained by each approach. Finally, Section 6.4 presents the results of the sensitivity analysis study, which is conducted by varying the number of available vehicles, their capacity, and the release time of the requests.

6.1 Scenarios Generation

To generate the instances, we used real data collected from the hospital. The data contains all the information about transportation requests received by the department of transportation

for one year. An instance generator is developed to create the instances, which have all known transport requests at the beginning of the day and those revealed afterwards. For each instance, the requests are ordered according to their arrival time.

Real-life instances are constructed directly from the requests performed during each day. For the stochastic scenarios, we analyzed the distribution of the number of requests for the seven days of the week. The instances were generated such that the number of requests is a random phenomenon that follows a Poisson distribution. Stochastic instances are created for each day of the week to resemble the actual operations.

6.2 Parameter Setting

This section is devoted to identifying the best values for the time bucket Δ, the time horizon of the stochastic approaches, the number of iterations η of the ALNS, and the number of scenarios to be used in the stochastic approaches. To conduct the tests, we randomly generated a set of 35 test instances consisting of five instances for each day of the week.

The first test determines the best value of the time bucket, which is used as an input time for the function $FreeDriver$ in Algorithms 3 and 4. This function verifies if there is a free driver within the time bucket Δ. We tested different time bucket values (*Bucket*) from 5 to 60 minutes. The average weighted lateness in minutes (*Late.*) and the average total travel time in minutes (*Travel*) are used as criteria to choose the best value for the time bucket. Note that the values of *Late.* and *Travel* are calculated as follows: we start by summing up the weighted lateness or travel time of all vehicles per day; we then calculate the average value per each month. Finally, the latter is used to calculate an average value for all months which is reported in all the tables of the paper. For the SBPA, we tested three configurations: SH-W, SH-D, and SH-GF, which correspond to the three relocation strategies *Wait*, *Depot*, and *Ground Floor*. Table 2 presents the results of the tests, which show that the best results are always obtained with a time bucket of five minutes. Hence, this value is adopted for all the subsequent tests.

The second test aims at choosing the time horizon for the stochastic approaches. We do not evaluate the static and the reoptimization heuristics because they are not affected by the time horizon. Table 3 presents the test results. We used the same evaluation criteria as in Table 2 and added the average total computational time for all instances in seconds (*Time*) as an additional criterion. The results show that the average total computational time increases significantly

Table 2: Alternative values for the time bucket

Bucket (min)	SH-W Late. (min)	SH-W Travel (min)	SH-W Time (s)	SH-D Late. (min)	SH-D Time (s)	SH-D Travel (min)	SH-GF Late. (min)	SH-GF Travel (min)	SH-GF Time (s)
5	**7.4**	**1,577.9**	**9.9**	**6.8**	**1,670.4**	**10.5**	**6.5**	**1,606.7**	**10.1**
10	8.3	1,632.0	9.7	7.5	1,686.2	10.3	7.5	1,638.3	10.3
20	8.1	1,655.7	9.4	8.1	1,719.9	10.1	7.9	1,681.9	10.4
30	9.8	1,662.9	9.7	9.0	1,717.1	10.0	9.7	1,671.6	10.0
40	10.3	1,651.8	9.4	9.2	1,709.6	10.1	8.6	1,680.3	9.2
50	9.0	1,641.0	9.3	8.3	1,694.4	10.5	7.2	1,664.3	9.2
60	7.2	1,637.2	10.5	8.4	1,690.4	10.8	8.9	1,665.1	9.7
Average	8.6	1,636.9	9.7	8.2	1,698.3	10.3	8.0	1,658.3	9.8

as the horizon increases. The smaller the horizon, the less computational time is needed. For SH-W and the B&R, the best results are obtained with a horizon of 5 minutes, while for SH-D and SH-GF, the best results are obtained with a horizon of 10 minutes. Considering the small trade-offs, for the next tests, we use time horizons of 10 minutes.

Table 3: Alternative values for the time horizon

Horizon	SH-W Late. (min)	SH-W Travel (min)	SH-W Time (s)	SH-D Late. (min)	SH-D Travel (min)	SH-D Time (s)	SH-GF Late. (min)	SH-GF Travel (min)	SH-GF Time (s)	B&R Late. (min)	B&R Travel (min)	B&R Time (s)
5	**7.0**	**1,572.2**	**6.2**	7.8	1,666.1	7.5	7.8	1,601.1	6.9	**3.9**	**1,566.5**	**19.6**
10	7.4	1,577.9	10.0	**6.8**	**1,670.4**	**10.6**	**6.5**	**1,606.7**	**10.1**	5.3	1,587.1	31.4
20	8.1	1,580.5	19.6	11.2	1,679.8	21.8	7.5	1,599.1	22.2	7.3	1,567.8	56.9
30	8.2	1,585.2	37.8	7.5	1,672.7	42.0	8.3	1,617.7	40.2	5.3	1,555.5	103.5
40	9.4	1,582.0	68.2	7.7	1,662.2	68.3	6.7	1,612.4	70.7	6.7	1,554.3	148.7
50	8.4	1,584.0	116.2	7.0	1,672.5	118.9	7.9	1,616.2	120.2	7.1	1,557.6	233.0
60	7.4	1,582.5	157.2	8.9	1,671.2	156.7	9.0	1,619.0	162.6	6.6	1,554.9	277.3
Average	8.0	1,580.6	59.3	8.1	1,670.7	60.8	7.7	1,610.3	61.8	6.0	1,563.4	124.3

Next, we evaluate the impact of the number of sample scenarios (*# Scen.*). Table 4 presents the results of the tests. All the stochastic approaches are considered and tested with the number of scenarios between 5 and 30. The results show that a higher number of scenarios increases the average total computational time, which grows almost linearly. The best results are obtained using 10 scenarios, making it the value for the rest of the tests.

Finally, we test different values for the number of ALNS iterations. The results are presented in Table 5. In this part of the parameter setting, we tested the static approach, the RH with its three configurations, the SBPA with its three configurations, and the B&R heuristic. Each of these algorithms has been executed with the number of ALNS iterations varying between 50 and 4000. The results show that increasing the number of iterations η improves the average travel time and increases the average total computational time. Meanwhile, the increase of η

Table 4: Alternative values for the number of scenarios

# Scen.	SH-W			SH-D			SH-GF			B&R		
	Late. (min)	Travel (min)	Time (s)	Late. (min)	Travel (min)	Time (s)	Late. (min)	Travel (min)	Time (s)	Late. (min)	Travel (min)	Time (s)
5	8.7	1,574.5	3.1	9.1	1,662.4	5.7	7.7	1,606.2	4.9	5.7	1,559.5	10.1
10	**7.1**	**1,572.2**	**6.2**	**6.8**	**1,670.4**	**10.6**	**6.5**	**1,606.7**	**10.1**	**3.9**	**1,566.5**	**19.6**
15	9.4	1,579.2	8.7	7.9	1,668.9	16.3	8.0	1,601.0	14.9	4.5	1,565.2	30.8
20	8.9	1,573.7	11.8	9.1	1,674.2	20.6	8.0	1,618.8	19.7	4.6	1,578.7	41.4
25	7.6	1,577.6	16.1	7.4	1,673.4	26.9	7.1	1,606.9	22.7	4.3	1,577.4	57.1
30	8.3	1,578.3	16.8	7.8	1,677.8	31.3	7.6	1,608.4	28.3	4.6	1,572.5	65.8
Average	8.3	1,575.9	10.5	8.0	1,671.2	18.6	7.5	1,608.0	16.8	4.6	1,569.9	37.5

does not always guarantee a decrease in the weighted lateness. We use 2000 ALNS iterations for the rest of the tests as it represents a good trade-off between the weighted lateness and computational time, which is about 35 seconds on average. For the RH, all three versions needed low computational time, finding a solution in about 2.4 seconds on average. The three versions of the SBPA could find a solution in an average of 40 seconds. Finally, the B&R was the slowest, with a computational time of 117.1 seconds.

Table 5: Alternative values for the number of ALNS iterations

Algorithm	50 ALNS iterations			250 ALNS iterations			500 ALNS iterations			1000 ALNS iterations			2000 ALNS iterations			4000 ALNS iterations		
	Late. (min)	Travel (min)	Time (s)	Late. (min)	Travel (min)	Time (s)	Late. (min)	Travel (min)	Time (s)	Late. (min)	Travel (min)	Time (s)	Late. (min)	Travel (min)	Time (s)	Late. (min)	Travel (min)	Time (s)
Static	1.4	1,454.2	1.3	0.8	1,420.6	3.7	0.2	1,387.1	8.0	0.2	1,359.2	16.8	0.1	1,348.2	35.1	0.1	1,332.6	69.8
RH-W	9.6	1,569.4	0.1	8.5	1,548.1	0.3	6.4	1,559.0	0.6	9.9	1,569.1	1.3	5.7	1,561.4	2.5	6.9	1,547.8	4.8
RH-D	9.6	1,673.0	0.2	6.6	1,661.6	0.4	8.1	1,657.3	0.7	6.2	1,661.6	1.3	8.0	1,672.9	2.6	6.6	1,656.4	5.2
RH-GF	9.3	1,588.9	0.1	7.2	1,575.7	0.3	8.1	1,573.8	0.6	7.7	1,584.3	1.2	7.9	1,577.9	2.2	6.3	1,569.5	4.3
SH-W	11.8	1,625.6	2.1	8.2	1,616.3	5.2	8.3	1,632.0	10.0	9.6	1,620.2	19.0	9.3	1,623.7	37.3	10.2	1,606.2	81.4
SH-D	10.6	1,695.1	2.3	7.6	1,689.4	5.5	7.5	1,686.2	10.6	8.2	1,692.5	20.7	5.9	1,691.1	42.1	6.7	1,675.0	89.4
SH-GF	7.9	1,642.3	2.2	7.3	1,634.3	4.6	7.5	1,638.3	11.4	9.6	1,647.4	18.3	8.6	1,645.5	40.7	6.2	1,630.0	78.8
B&R	5.7	1,575.4	7.0	5.3	1,563.7	15.8	5.3	1,587.1	31.0	4.8	1,569.4	62.3	6.6	1,559.4	117.1	5.1	1,570.4	245.7
Average	8.2	1,603.0	1.9	6.4	1,588.7	4.5	6.4	1,590.1	9.1	7.0	1,588.0	17.6	6.5	1,585.0	35.0	6.0	1,573.5	72.4

6.3 Computational results

In this section, we present the results obtained by each one of the proposed methods. Table 6 summarizes the results of the tests for all instances, where for each algorithm, the table reports the average weighted lateness in minutes, the average travel time in minutes, the average waiting (parked) time of the vehicles in minutes (*Waiting*), the average computational time per event *T/E*, and the last column describes the average total computational time in seconds for all instances. Note that *Waiting* is calculated in a similar way as *Late.* and *Travel*, and the average number of events for all instances is 140.3.

As expected, the static approach yields the best possible results by having an average weighted lateness of only 2.1 minutes, an average travel time of 1305.2 minutes, and an average runtime

of 32.6 seconds. These results represent a lower bound on the other methods' solutions, which tend to perform worse than the static approach, given that they operate under actual conditions with uncertainty and dynamism.

The RH with its three configurations is the fastest approach since it optimizes small subproblems whenever new requests arrive. It can find a solution in less than three seconds. In this approach, the requests are processed according to their release time. Therefore, past decisions are locked at each iteration, and the algorithm processes only the available requests at the time t. Comparing the three configurations of the RH, one can notice that RH-W has the lowest travel time. This is logical because vehicles must wait for new requests at their last delivery location, preventing unnecessary traveling at the expense of potential patient waiting time. The second configuration, RH-D, yields the lowest weighted lateness but still has the highest travel time because drivers must return to the depot and wait for new requests each time. The RH-GF yields a relatively similar value of the weighted lateness to that of the RH-D (increased by 0.6%), and the travel time is better than the one given by the RH-D but slightly worse than the one given by the RH-W.

The three configurations of the SPBA approach are tested using 10 stochastic scenarios in addition to the 365 real scenarios. Similar to the case of the RH, the second strategy, SH-D, yields the best results in terms of weighted lateness and waiting time, but the worst travel time. Moreover, the comparison between RH and SBPA shows that incorporating demand forecast yields a positive impact on the weighted lateness of the SH-D. However, the impact is limited in the other configurations.

Finally, the B&R reduced the weighted lateness by 28.8% and the travel time by 6.2% on average compared to the previous dynamic and stochastic approaches. However, the average total computational time increased by 82%. Overall, regarding the quality of the results, one can conclude that the B&R is the best approach compared to the RH and the SBPA approaches.

Figure 2 depicts the average monthly weighted lateness for the four approaches (static, RH-D, SH-D, and B&R). One can notice that the static approach has the best performance regarding the weighted lateness over the 12 months of the year. Considering the other approaches, one can observe that the B&R has the best performance most of the time. In fact, except for the third and fifth months, the B&R can find solutions with lower weighted lateness. The SH-D slightly outperforms B&R in the third month and had higher weighted lateness than the RH heuristic in

Table 6: Summary of the results of all methods

Algorithm	Late. (min)	Travel (min)	Waiting (min)	T/E (s)	Time (s)
Static	2.1	1,305.2	372.7	0.23	32.6
RH-W	35.9	1,526.0	313.6	0.02	2.4
RH-D	34.9	1,621.0	283.9	0.02	2.8
RH-GF	35.1	1,542.4	317.3	0.02	2.4
SH-W	40.8	1,541.8	312.5	0.18	25.2
SH-D	33.5	1,626.5	295.6	0.33	45.7
SH-GF	36.1	1,563.2	323.5	0.27	38.2
B&R	26.6	1,531.3	337.1	0.59	83.1

some months. The RH-D has a higher weighted lateness compared to the previous approaches. However, it performed better than the SH-D in some months and slightly better than the B&R in the fifth month.

Figure 2: Average monthly weighted lateness

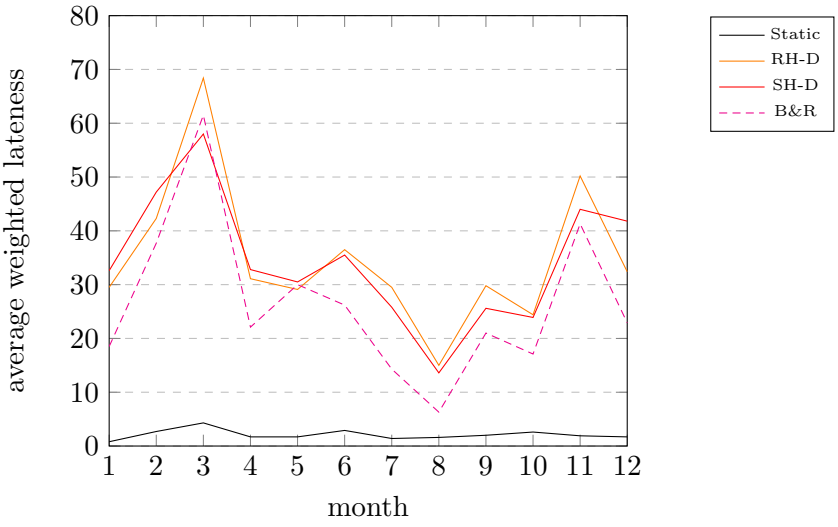
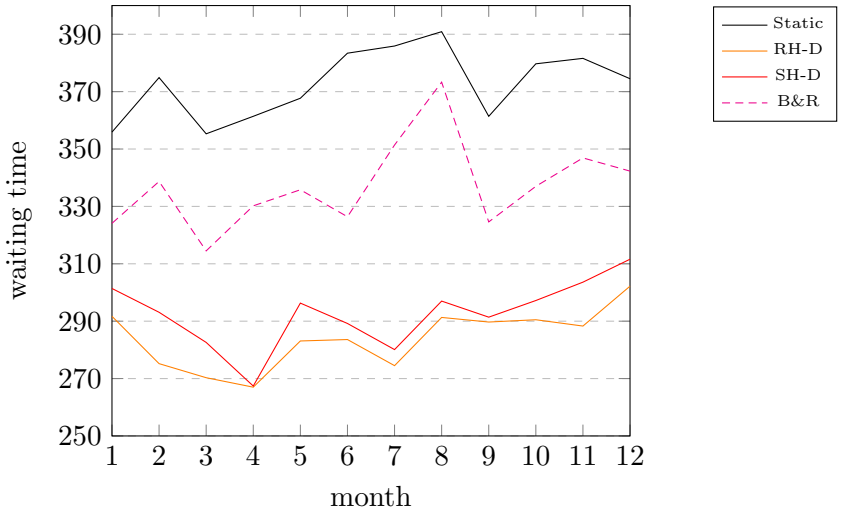


Figure 3 shows the average monthly waiting time. One can notice a correlation between weighted lateness and waiting time. The waiting time is inversely correlated with the weighted lateness for all the algorithms. The static algorithm has the highest waiting time for all the months; next is the B&R, then the SH-D, and last is the RH-D. Figure 3 indicates that the B&R has a higher waiting time than the SH-D and the RH-D, highlighting that the B&R performs the best relocations. In addition, it shows that using the stochastic information to relocate the vehicles has a significant impact on the results.

Figure 3: Average monthly waiting time



6.4 Sensitivity Analysis

This section analyzes the impact of varying the number of vehicles, their capacity, and the requests' release time on the results of the proposed approaches. We start by evaluating the impact of varying the number of vehicles of each type. The tests are performed using all 365 real instances. Table 7 summarizes the results of using different values of the number of ambulances with the different algorithms. We varied this number from one to six. The results show that using only one ambulance instead of two (default case) significantly impacts the results, as the weighted lateness increases from an average of 30.6 minutes for all the approaches to an average of 366.7 minutes. Moreover, the travel time increased by 9.2%. Adding one more ambulance to a total of three further improves the weighted lateness and the travel time. Adding additional ambulances negatively impacts the travel time while the weighted lateness continues the improvement trend.

Table 7: Sensitivity analysis regarding the number of ambulances

Algorithm	1 amb & 3 EV		2 amb & 3 EV		3 amb & 3 EV		4 amb & 3 EV		5 amb & 3 EV		6 amb & 3 EV	
	Late. (min)	Travel (min)	Late. (min)	Travel (min)	Late. (min)	Travel (min)	Late. (min)	Travel (min)	Late. (min)	Travel (min)	Late. (min)	Travel (min)
Static	87.9	1,499.4	2.1	1,305.2	1.3	1,254.6	1.1	1,246.9	1.1	1,254.8	1.1	1,262.0
RH-W	418.0	1,657.3	35.9	1,526.0	11.6	1,516.7	6.5	1,523.0	5.1	1,516.8	3.5	1,512.7
RH-D	406.4	1,750.6	34.9	1,621.0	8.1	1,590.5	3.9	1,584.6	3.2	1,586.8	3.1	1,587.9
RH-GF	403.6	1,676.1	35.1	1,542.4	8.0	1,542.6	4.4	1,559.8	3.3	1,556.2	3.1	1,554.0
SH-W	425.5	1,676.4	40.8	1,541.8	11.7	1,521.3	6.3	1,519.3	4.7	1,514.1	3.7	1,509.8
SH-D	408.6	1,754.3	33.5	1,626.5	8.3	1,596.1	3.9	1,592.4	3.0	1,590.6	3.0	1,594.7
SH-GF	396.9	1,691.9	36.1	1,563.2	8.3	1,547.4	4.5	1,557.8	3.3	1,553.5	3.1	1,551.9
B&R	386.5	1,691.3	26.6	1,531.3	5.2	1,511.9	3.4	1,529.3	2.7	1,563.6	2.2	1,565.1
Average	366.7	1,674.7	30.6	1,532.2	7.8	1,510.1	4.2	1,514.1	3.3	1,517.1	2.9	1,517.3

The results of Table 8 demonstrate the same trend as in Table 7. Increasing the number of electric vehicles significantly impacts the weighted lateness. An improvement of the travel time is also observed but not as significant as the weighted lateness. Moreover, one may observe that the travel time slightly increased when the number of electric vehicles increased from one to two.Tables 7 and 8 show that overall, increasing the fleet size positively impacts the solutions quality. Furthermore, one may observe that ambulances have more impact on the results than electric vehicles. One can also observe that when the number of vehicles decreases, the third strategy yields better results than the first and second strategies in both the RH and the SBPA approaches. Moreover, when the number of electric vehicles increases, the third strategy can find a better solution than the others. Finally, regardless of the type of vehicle to increase or decrease, the B&R always finds better solutions than the RH and SBPA approaches.

Table 8: Sensitivity analysis regarding the number of electric vehicles

Algorithm	2 amb & 1 EV		2 amb & 2 EV		2 amb & 3 EV		2 amb & 4 EV		2 amb & 5 EV		2 amb & 6 EV	
	Late. (min)	Travel (min)	Late. (min)	Travel (min)	Late. (min)	Travel (min)	Late. (min)	Travel (min)	Late. (min)	Travel (min)	Late. (min)	Travel (min)
Static	488.7	1,434.3	24.1	1,358.9	2.1	1,305.2	1.4	1,289.7	1.5	1,284.1	1.4	1,281.1
RH-W	989.4	1,539.5	146.0	1,550.2	35.9	1,526.0	22.8	1,513.5	19.2	1,509.3	17.3	1,509.6
RH-D	987.5	1,552.8	137.1	1,610.3	34.9	1,621.0	22.4	1,626.2	18.6	1,624.8	19.1	1,629.3
RH-GF	985.2	1,543.8	134.0	1,555.3	35.1	1,542.4	19.7	1,532.4	16.6	1,530.6	15.1	1,531.2
SH-W	1,011.7	1,540.9	143.7	1,556.6	40.8	1,541.8	23.6	1,529.0	19.3	1,524.5	17.0	1,521.2
SH-D	1,020.0	1,553.9	130.0	1,610.8	33.5	1,626.5	22.1	1,633.7	19.2	1,633.0	19.1	1,633.8
SH-GF	1,018.1	1,545.8	133.6	1,567.0	36.1	1,563.2	20.8	1,551.8	16.2	1,556.5	16.0	1,555.8
B&R	978.7	1,555.2	109.0	1,553.1	26.6	1,531.3	14.4	1,518.5	11.8	1,518.9	11.2	1,517.6
Average	934.9	1,533.3	119.7	1,545.3	30.6	1,532.2	18.4	1,524.4	15.3	1,522.7	14.5	1,522.5

Next, we analyze the capacity of the vehicles. As is relevant in healthcare transportation, we highlight the case where restrictions are imposed to limit the risk of contamination, such as the COVID-19. Under this scenario, the hospital must adapt to the new safety measures, including social distancing. Hence, the capacity is reduced to one person per vehicle. Technically, only electric vehicles are affected by this measure since the maximum capacity of the ambulances is one person by default. Table 9 presents the sensitivity analysis results with respect to the capacity of the vehicles. The first thing to observe is that limiting the capacity of electric vehicles yields an increase by almost 60% of the weighted lateness to attend an average of 48.9 minutes, and the travel time slightly increased by 4.3% on average. In contrast, the average total computational time and waiting time decreased by 8.4% and 4.6%, respectively.

Finally, we evaluate the impact on the solutions when we vary the release time to understand the impact of having more information in advance. The release time is set to some minutes before the request can be served. This means that the information about the request becomes known

Table 9: Vehicle capacity analysis

Algorithm	Cap amb = 1 & Cap EV = 1			Cap amb = 1 & Cap EV = 4		
	Late. (min)	Travel (min)	Waiting (min)	Late. (min)	Travel (min)	Waiting (min)
Static	4.3	1,428.0	357.0	2.1	1,305.2	372.7
RH-W	61.0	1,591.7	296.1	35.9	1,526.0	313.6
RH-D	55.5	1,670.4	274.2	34.9	1,621.0	283.9
RH-GF	54.6	1,604.6	302.1	35.1	1,542.4	317.3
SH-W	64.1	1,602.8	299.9	40.8	1,541.8	312.5
SH-D	55.4	1,674.5	280.3	33.5	1,626.5	295.6
SH-GF	54.9	1,619.8	307.0	36.1	1,563.2	323.5
B&R	41.2	1,598.1	320.6	26.6	1,531.3	337.1
Average	48.9	1,598.8	304.7	30.6	1,532.2	319.5

a few minutes before its time window opens. We evaluated all algorithms with 0, 10, 20, 40, and 60 minutes of release time values. Table 10 presents the results showing that increasing the release time up to 40 minutes yields a significant positive impact on the quality of the solutions in terms of weighted lateness and travel time, which highlights the added value of knowing the information about the requests in advance. However, no improvements are observed by setting the release time to more than 40 minutes, which is an important information in practice. Moreover, one may also observe that the B&R still provides the best results compared to the RH and SBPA approaches in all scenarios of the release time values. This also confirms its excellent performance as was already demonstrated.

Table 10: Comparison for different release times

Algorithm	Release time = 0		Release time = 10		Release time = 20		Release time = 40		Release time = 60	
	Late. (min)	Travel (min)	Late. (min)	Travel (min)	Late. (min)	Travel (min)	Late. (min)	Travel (min)	Late. (min)	Travel (min)
Static	2.1	1,305.2	1.3	1,277.4	1.1	1,278.8	1.3	1,277.8	1.3	1,277.8
RH-W	35.9	1,526.0	14.7	1,443.4	6.3	1,387.8	3.7	1,313.9	3.7	1,313.9
RH-D	34.9	1,621.0	16.3	1,511.1	6.2	1,429.3	3.8	1,331.4	3.8	1,331.4
RH-GF	35.1	1,542.4	14.6	1,456.6	6.3	1,393.7	3.7	1,315.9	3.7	1,315.9
SH-W	40.8	1,541.8	13.7	1,444.9	7.1	1,384.3	4.2	1,313.0	4.2	1,313.0
SH-D	33.5	1,626.5	14.4	1,508.9	6.5	1,427.0	4.3	1,332.9	4.3	1,332.9
SH-GF	36.1	1,563.2	13.9	1,457.5	7.3	1,389.4	4.1	1,317.8	4.1	1,317.8
B&R	26.6	1,531.3	11.4	1,436.5	7.0	1,380.4	3.3	1,310.6	3.3	1,310.6
Average	30.6	1,532.2	12.5	1,442.0	6.0	1,383.8	3.5	1,314.2	3.5	1,314.2

7 Conclusion

In this paper, we studied a dynamic pickup and delivery problem with soft time windows based on a real case study of a hospital in Italy, composed of several buildings connected by external roads and underground tunnels. This setting is characterized by high complexity due to uncertainty and a high number of daily requests to move patients from one location to another using a heterogeneous fleet of vehicles. In this healthcare facility, the decision-maker relies on experience to manage the transportation flow. We aim to improve the efficiency of these transportation services by minimizing the total weighted lateness and travel time.

We proposed an offline algorithm that has full information available, serving as an oracle providing the best possible solutions. We then solve the dynamic problem using reoptimization heuristics that benefit from vehicle relocation strategies: waiting at the last location, moving back to the depot, or moving to the ground floor of the last visited building. The three versions of the dynamic algorithm do not incorporate stochastic information about future events. Furthermore, we introduced a dynamic and stochastic algorithm based on the scenario-based planning approach. As in the previous approach, we developed three versions to evaluate the impact of the relocation strategies. A consensus function determines where the subsequent request is more likely to appear. Finally, the last approach is also dynamic and stochastic, in the form of a branch-and-regret algorithm. This algorithm uses stochastic scenarios and relocates each free driver to the location that generates the best average future expected cost.

We evaluated our solution approaches using real data from our industrial partner. The results show that overall, the B&R approach outperforms the other approaches; it positively impacts on the efficiency of the hospital operations and improves the service quality by reducing the weighted lateness. These findings highlight the importance and the added value of incorporating stochastic information in the optimization process. We also conducted a detailed sensitivity analysis to evaluate the performance of the proposed approaches when varying some problem parameters, namely the number and the capacity of the vehicles, and the release time of the requests. The results show the fleet size significantly impacts the weighted lateness, limiting the capacity of the electric vehicles due to health restrictions deteriorates the quality of the solutions, and having more information some minutes in advance significantly helps find better solutions regarding waiting time and travel time.

Acknowledgments

This work was partly supported by the Canadian Natural Sciences and Engineering Research Council (NSERC) under grants 2019-00094 and 2021-04037. We thank Compute Canada for providing high-performance parallel computing facilities.

References

[1] I. Aziez, J.-F. Côté, and L.C. Coelho. Fleet sizing and routing of healthcare automated guided vehicles. *Transportation Research Part E: Logistics and Transportation Review*, 161:102679, 2022.

[2] A. Beaudry, G. Laporte, T. Melo, and S. Nickel. Dynamic transportation of patients in hospitals. *OR Spectrum*, 32(1):77–107, 2010.

[3] R. Bent and P. Van Hentenryck. Waiting and relocation strategies in online stochastic vehicle routing. In *IJCAI*, volume 7, pages 1816–1821, 2007.

[4] R.W. Bent and P. Van Hentenryck. Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research*, 52(6):977–987, 2004.

[5] J.-F. Côté, T. Alves de Queiroz, F. Gallesi, and M. Iori. A branch-and-regret algorithm for the same-day delivery problem. *Transportation Research Part E: Logistics and Transportation Review*, 177:103226, 2023.

[6] M. Gendreau, F. Guertin, J.-Y. Potvin, and É. Taillard. Parallel tabu search for real-time vehicle routing and dispatching. *Transportation Science*, 33(4):381–390, 1999.

[7] T. Hanne, T. Melo, and S. Nickel. Bringing robustness to patient flow management through optimized patient transports in hospitals. *Interfaces*, 39(3):241–255, 2009.

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

[9] L.M. Hvattum, A. Løkketangen, and G. Laporte. Solving a dynamic and stochastic vehicle routing problem with a sample scenario hedging heuristic. *Transportation Science*, 40(4):421–438, 2006.

[10] L.M. Hvattum, A. Løkketangen, and G. Laporte. A branch-and-regret heuristic for stochastic and dynamic vehicle routing problems. *Networks*, 49(4):330–340, 2007.

[11] Y. Kergosien, C. Lente, D. Piton, and J.-C. Billaut. A tabu search heuristic for the dynamic transportation of patients between care units. *European Journal of Operational Research*, 214(2): 442–452, 2011.

[12] S. Landry and R. Philippe. How logistics can service healthcare. *Supply Chain Forum: An International Journal*, 5(2):24–30, 2004.

[13] E. Melachrinoudis, A.B. Ilhan, and H. Min. A dial-a-ride problem for client transportation in a health-care organization. *Computers & Operations Research*, 34(3):742–759, 2007.

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

[15] S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, 2006.

[16] Y. Song, M.W. Ulmer, B.W. Thomas, and S.W. Wallace. Building trust in home services—stochastic team-orienteering with consistency constraints. *Transportation Science*, 54(3):823–838, 2020.

[17] M.W. Ulmer, J.C. Goodson, D.C. Mattfeld, and B.W. Thomas. On modeling stochastic dynamic vehicle routing problems. *EURO Journal on Transportation and Logistics*, 9(2):100008, 2020.

[18] UNPD. La prochaine frontière: le développement humain et l'anthropocène, 2020. URL `http://hdr.undp.org/sites/all/themes/hdr_theme/country-notes/fr/CAN.pdf`.

[19] F. Y. Vincent, M. M. Zegeye, S. G. Gebeyehu, P. A. Y. Indrakarna, and P. Jodiawan. A matheuristic algorithm for the share-a-ride problem. *Expert Systems with Applications*, page 120569, 2023.

[20] F. Y. Vincent, M. M. Zegeye, S. G. Gebeyehu, P. A. Y. Indrakarna, and P. Jodiawan. The multi-depot general share-a-ride problem. *Expert Systems with Applications*, 213:119044, 2023.

[21] S.A. Voccia, A.M. Campbell, and B.W. Thomas. The same-day delivery problem for online purchases. *Transportation Science*, 53(1):167–184, 2019.

[22] The world bank. Current health expenditure, 2018. URL `data.worldbank.org`.