Comparison of anticipatory algorithms for a dial-a-ride problem

Ulrike Ritzinger, Jakob Puchinger, Christian Rudloff, Richard F. Hartl

European Journal of Operational Research 301 (2022) 591–608

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Production, Manufacturing, Transportation and Logistics

Comparison of anticipatory algorithms for a dial-a-ride problem

Ulrike Ritzinger[a,*], Jakob Puchinger[b,c], Christian Rudloff[a], Richard F. Hartl[d]

[a] AIT Austrian Institute of Technology GmbH, Center for Energy, Giefinggasse 2, Vienna 1210, Austria
[b] Université Paris-Saclay, CentraleSupélec, Laboratoire Génie Industriel, Gif-sur-Yvette, 91190, France
[c] Institut de Recherche Technologique SystemX, Palaiseau 91120, France
[d] University of Vienna, Department of Business Decisions and Analytics, Oskar-Morgenstern-Platz 1, Vienna 1090, Austria

ARTICLE INFO

Article history:
Received 17 February 2021
Accepted 30 October 2021
Available online 5 November 2021

Keywords:
Transportation
Dial-a-Ride problem
Dynamic
Stochastic
Real-world application

ABSTRACT

Progress in digitalization opens opportunities to capture accurate transportation logistics data and provide advanced decision support, which leads into the question of how to efficiently exploit this progress in order to improve solution quality in transportation services. Here we address this issue in the context of a dynamic and stochastic patient transportation problem, where besides considering new events, we also incorporate stochastic information about future events. We propose different anticipatory algorithms and investigate which algorithm performs best according to the given settings in a real-world application. We therefore address different types of dynamic events, appropriate response times, and the synchronization of real-world data with the plan. In order to test and analyze how the algorithms behave and perform, we apply the concept of a digital twin. The implemented anticipatory algorithms compared here are a sample scenario planning approach and two waiting strategies. The question of the value of more sophisticated algorithms compared to algorithms with less computational effort is investigated. The experimental results show that solution quality benefits from incorporating information about future requests, and that simple waiting strategies prove most suitable for such a highly dynamic environment. We find that in highly stochastic environments, a rescheduling should be done whenever a new event occurs, whereas in less stochastic environments it is better to let the optimization engine run a bit longer and not start reoptimization after every new event.

1. Introduction

Recent advances information technology (detailed tracking information, efficient data storage, mobile communications, etc.) provide a strong foundation for extensive data collection and real-time support, and the allied increase in computing power facilitates improvements in solution quality by processing this data. Research attention in the area of transportation problems has recently turned to focus on dynamic and stochastic approaches, where besides considering dynamic events, information about possible future events is incorporated during the solution-finding process. The focus of research is mainly on obtaining better results by incorporating stochastic information compared to the pure dynamic counterpart. Even though most dynamic and stochastic solution approaches show that solution quality benefits from incorporating stochastic information, it usually comes at a cost of higher computational and implementational effort. However, modern decision support systems demand real-time decision-making, which requires fast reactions and responses. Crainic, Gendreau, & Potvin (2009) assert that, there is a gap between state-of-the-art optimization methods and their applications in real-time decision support systems. A crucial advance to close this gap would be to provide solutions where the time for computational calculations is appropriate to the dynamics of the given system. Little research has attempted to compare more sophisticated solution approaches against approaches that are less costly in terms of implementation and computation effort. Research questions thus arise on the value of applying a complex approach instead of using approaches requiring less time and effort, and on the suitability and applicability of different approaches for various dynamic factors.

In this work, we present different anticipatory algorithms for a patient transportation problem and analyze their performance. We investigate which algorithm performs best according to the given settings in a real-world application, such as different dynamic events, short response times, and the synchronization of

* Corresponding author.
E-mail addresses: ulrike.ritzinger@ait.ac.at (U. Ritzinger), jakob.puchinger@irt-systemx.fr (J. Puchinger), christian.rudloff@ait.ac.at (C. Rudloff), richard.hartl@univie.ac.at (R.F. Hartl).

https://doi.org/10.1016/j.ejor.2021.10.060
0377-2217/

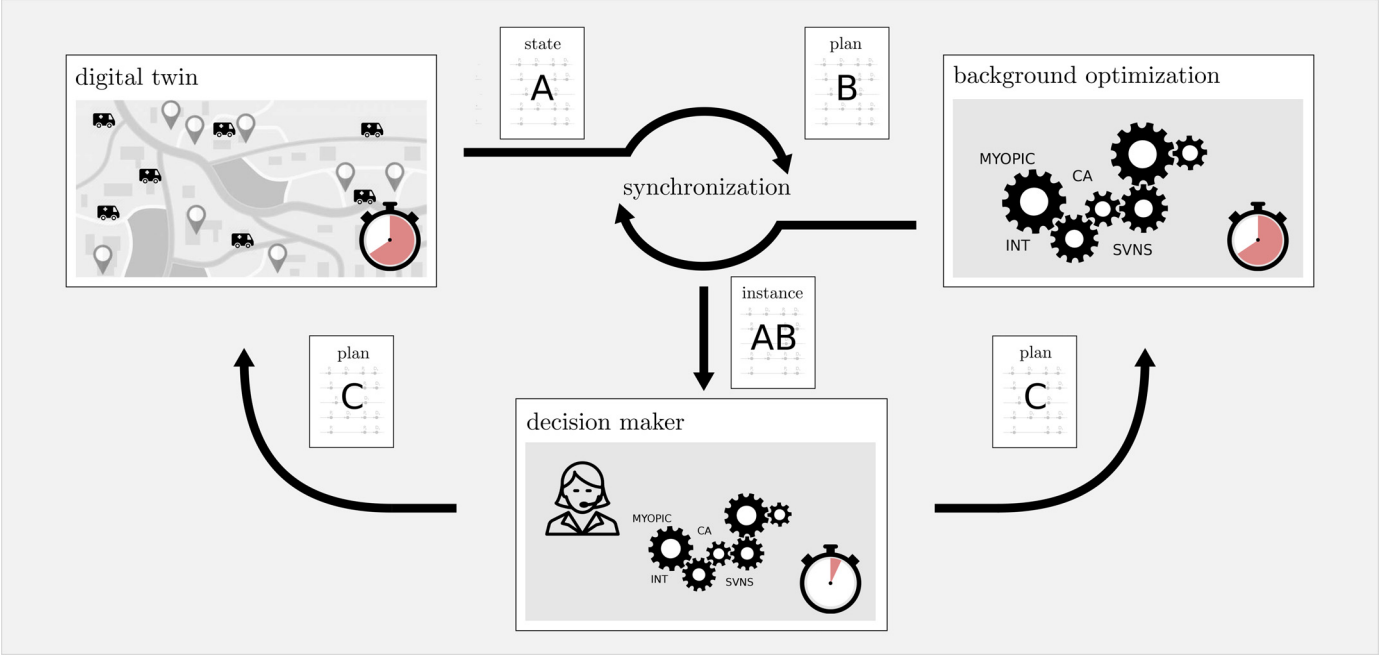


Fig. 1. System overview of a digital twin-based framework. Whenever a new event happens, the current state of the digital twin (A) and the best result from the background optimization (B) are synchronized and the appropriate instance (AB) is handed to the decision-maker. After considering the new event information and replanning (short response time), the updated plan (C) is sent to both the digital twin and the background optimization.

real-world data with planning data. Furthermore, we demonstrate that besides the classic *degree of dynamism* introduced by Lund, Madsen, & Rygaard (1996), there are other crucial aspects that determine the dynamism in such a system. This dynamism is an essential factor because it dictates the frequency of replannings and adjustments and thus the computational time of the optimization algorithms. For the algorithmic comparison, we develop two types of optimization approaches that incorporate stochastic information about future events. The first type implements a *waiting strategy* while the second type is a sample scenario approach. The waiting strategy determines suitable locations along a route for waiting, with the aim of positioning vehicles to areas where new customer requests are likely to arise. This can be done by applying a priori rules like *drive first* or *wait first*, or by incorporating stochastic information about future events, as done here. In a *sample-scenario approach*, a sample that includes scenarios with possible future events is generated at any decision point and then used as known information in the optimization algorithm. Furthermore, the performance of the algorithms is evaluated by considering different dynamic scenarios, such as dynamic requests and dynamic service end-times. To guarantee a fair comparison, the information about future events is obtained from a stochastic model and the same basic optimization algorithm is used for all approaches.

To test the performance of the optimization algorithms in a real-world environment, we implement a digital twin-based framework. The digital twin mirrors the real-world framework (vehicle movements and actions, service and travel times, etc.) and enables practical development, testing and analysis of our solution approaches. In order to obtain high solution quality despite short response times, a background optimization process is integrated. Whenever dealing with a dynamic real world, it is necessary to have a form of data synchronization between the current state and the optimization process. In the case studied here, where a background optimization process is implemented, it is necessary to synchronize the current system state and the optimized route plan before moving on to consider new events and replanning steps. Fig. 1 gives a schematic illustration of the system and the flow when a dynamic event happens. Whenever a dynamic event causes a reoptimization, the background optimization is interrupted and the best solution is synchronized with the current system state gathered from the digital twin. The output of the synchronization process is an instance representing the current situation (including new events) for the optimization algorithms. After the reoptimization, which reacts to the new data and very quickly provides a solution, decisions are made based on the best solution and then propagated back to the digital twin as well as to the background optimization.

In this work, we make several key contributions to the field of research into dynamic and stochastic transportation problems:

- We introduce anticipatory algorithms for a real-world dial-a-ride problem consisting of a large set of patient requests and multiple vehicles. The algorithms vary in their computational complexity, and the aim is to investigate the performance of more sophisticated approaches compared to approaches requiring less computational effort in a highly dynamic real-world environment. Furthermore, we investigate how the performance of the different anticipatory algorithms is affected when the dynamism in the system increases (e.g. deviation of planned times).
- We present different reoptimization strategies and demonstrate their impact on solution quality and the value of applying an appropriate reoptimization strategy for the given problem.
- Since the focus is on a real-world application, we propose a digital twin-based framework that enables sophisticated analysis of the performance of the algorithms and reoptimization strategies. Furthermore, we address the issue of communication flow and data synchronization between real-world environment and decision-makers. This is a key issue when dealing with dynamic vehicle routing problems, but it is often neglected in the literature.

Furthermore, the key experimental findings in this work are:

- Implementing the background optimization and the required synchronization process is worth the effort because results show an average 50% improvement on the primary objective.

- The frequency of calling reoptimization procedures is a crucial factor because it has an impact on the runtime of the background optimization. Results show that strategies involving more frequent reoptimizations yield better results when the number of dynamic events increases.
- The experimental results on the comparison of different anticipatory algorithms show that on aggregate, the waiting strategies yield better results than the sophisticated sample-scenario approach.

This article continues with a review of relevant literature in Section 2 and a detailed description of the problem in Section 3. Section 4 describes the basic optimization algorithm and its anticipatory adaptations in detail. Section 5 includes a description of the instances generated from real-world data, the parameter settings, and the results of the performance comparison between the algorithms developed. Finally, Section 6 summarizes our work and gives perspectives for future work.

2. Literature review

Dynamic and stochastic transportation problems have gained increasing research attention in the past few years. As our work considers multiple aspects (like data availability, modeling, system design, real-life application, anticipatory algorithms, and the problem class), here we review the most significant contributions.

2.1. Available information and uncertainty

Vehicle routing problems (VRPs) can be categorized based on available information and uncertainty, as introduced in Schorpp (2010) and utilized in Pillac, Gendreau, Guéret, & Medaglia (2013). The first category is the *static VRP*, which has different formulations presented in Toth & Vigo (2001), Cordeau, Laporte, Savelsbergh, & Vigo (2007), Laporte (2007) and Laporte (2009). The second category, *dynamic VRPs (DVRP)* is a well-studied category where not all information (mostly customer requests) is known in advance. The *degree of dynamism* introduced by Lund et al. (1996) and refined in Larsen, Madsen, & Solomon (2002) expresses the dynamics in a system as a ratio of dynamic requests to total number of requests. Variants of the degree of dynamism are discussed, including information about time windows and *reaction time* (temporal distance between the time a request becomes known and the end of the time window) as well. Broad reviews on DVRPs can be found in Psaraftis (1995), Jaillet & Wagner (2008), Pillac et al. (2013), and Psaraftis, Wen, & Kontovas (2016). As shown in Psaraftis et al. (2016), the scholarship mainly considers dynamic customers, followed by dynamic travel times and vehicle breakdown, but very little research has addressed dynamic end times of services (Yan et al. (2019)). In many real-world applications, one or several parameters are uncertain. The third category is *stochastic VRPs*, which deal with uncertain information about future events (e.g. stochastic demands, requests, travel times, or service times). A literature review is given in Oyola, Arntzen, & Woodruff (2017), a review of recent advances and future directions can be found in Gendreau, Jabali, & Rei (2016), and a unified framework for stochastic optimization is presented in Powell (2019). The fourth category is the *dynamic and stochastic VRP*, considered here. This type of VRP is now attracting greater research attention, as is has the advantage of efficiently handling dynamic events while also incorporating stochastic information based on previously revealed data into the solution approaches. The survey by Ritzinger, Puchinger, & Hartl (2016b) summarizes recent literature in this area and analyzes the difference in solution quality when considering purely dynamic or stochastic approaches compared to approaches that consider dynamic and stochastic aspects together.

2.2. Modeling dynamic and stochastic information

The increasing availability of data and computing power demands new views and models for decision support(Savelsbergh & Van Woensel (2016)) and brings new challenges in terms of modeling dynamic events while at the same time incorporating stochastic information about future events (Speranza (2018)). Ulmer, Goodson, Mattfeld, & Thomas (2020) recently provided a modeling framework connecting applications and methods for dynamic and stochastic VRPs. There is growing interest in *same-day delivery problems*, which is a dynamic routing problem where customer requests arise during the day and vehicles have to pick up the goods at the depot before serving the request and where information about future requests is incorporated in order to make better decisions. Voccia, Campbell, & Thomas (2019) propose a solution method based on a *multiple scenario approach* as introduced in Bent & Van Hentenryck (2004) for a multi-vehicle dynamic pickup and delivery problem, and Ulmer, Mattfeld, & Köster (2018) implement an *approximate dynamic programming (ADP)* algorithm (Powell, Simao, & Bouzaiene-Ayari (2012)) where immediate and future impact is considered as an input to decision-making for single-vehicle routing. Similar to this problem is the *dynamic dispatching problem*, as investigated in Van Heeswijk, Mes, & Schutten (2019) via an ADP algorithm and in Klapp, Erera, & Toriello (2018) who applied a rollout algorithm of an a priori policy. These approaches are computationally intensive and applied on single-vehicle problems or relatively small instances and are therefore unsuitable for large-scale instances like in our case.

2.3. System design

An important task when dealing with dynamic problems is the design and implementation of a system that is capable of handling dynamic events and evaluating the performance of the algorithms. One possibility is to implement an *event-driven* system where predefined events cause a reoptimization (start a decision epoch) such as in Steever, Karwan, & Murray (2019), Bent & Van Hentenryck (2004), and Schilde, Doerner, & Hartl (2011). Another possibility is to define periodic decision points for updating operations (Klapp et al., 2018; Najmi, Rey, & Rashidi, 2017). One factor in preferentially choosing a *time-driven* or event-driven implementation lies in the expected response time of the system. For quick reactions to an event, the better way is to implement an event-driven approach, because periodical reoptimization may imply a longer lag to reaction. Pillac, Guéret, & Medaglia (2012b) distinguish between *periodic reoptimization* and *continuous reoptimization*, where periodic reoptimization approaches start with a solution, and updates are performed whenever a new optimization trigger is given. To minimize response time to the dispatcher optimization algorithms with short computational times are needed, as proposed in Lin et al. (2014), Najmi et al. (2017) and Steever et al. (2019). Continuous reoptimization approaches perform optimization throughout the day (background optimization) and store solutions in an adaptive memory (or solution pool) and the information from the memory is aggregated whenever a reoptimization is triggered. Such approaches are presented in Gendreau, Guertin, Potvin, & Taillard (1999) and Bent & Van Hentenryck (2004). However, the in-memory solutions have to be coherent with the current system state. The system design and the communication between the optimization and the real-world configuration are often neglected in the literature. For example, some approaches stop the clock for the reoptimization procedure, whereas in the real world, the clock continues. Furthermore, the models need to specify what happens if another event arises in the meantime. In this work, we address these points and specify how the communication and synchronization between the system and the real world can be handled. A fur-

ther research question investigated in this work is the impact of different reoptimization strategies with respect to solution quality. To the best of our knowledge, the impact of frequency of reoptimization is only considered in Archetti, Feillet, Mor, & Speranza (2020).

2.4. Real-time vehicle routing

A review on solution concepts and algorithms for real-time vehicle routing is given in Ghiani, Guerriero, Laporte, & Musmanno (2003). Decision support systems for dynamic routing problems can be found in Attanasio, Bregman, Ghiani, & Manni (2007), Pillac, Guéret, & Medaglia (2012a) and Lin et al. (2014). Attanasio et al. (2007) consider a dynamic stochastic real-time fleet management problem where a travel time and demand forecast module and a job allocation module are presented. The problem is implemented as an event-based approach with parallel solution algorithms, and a background optimization running when there are no dynamic events to handle. In contrast to Lin et al. (2014), where the decision support system aims to deliver quick responses because the call center has to reply to the customers immediately, Pillac et al. (2012a) designed an event-driven framework enabling flexible online optimization (parallelized) for a real-time VRP with stochastic demands. Even though there are some very interesting approaches, there is still a gap between state-of-the-art optimization and embedding it in real-life decision support systems (Crainic et al., 2009). However, technology advancements have recently spurred increasing attention to the digital twin concept across various industries (Cimino, Negri, & Fumagalli, 2019). A digital twin is a digital representation of physical objects or a system and decision support is an important use case for digital twins in logistics systems. For example, Korth, Schwede, & Zajac (2018) propose a digital twin for real time management of logistics systems, while Greif, Stein, & Flath (2020) present digital twins for construction site logistics. Recent reviews on this topic are given in Jones, Snider, Nassehi, Yon, & Hicks (2020) and Liu, Fang, Dong, & Xu (2020). Existing digital twins are mainly introduced for production and logistic systems, so to provide a real-time decision support system, we introduce a novel digital twin-based framework for transportation problems.

2.5. Anticipatory algorithms

One contribution of this work is that we explore the benefit of sophisticated algorithms compared to approaches requiring less computational and implementational effort for real-world applications. We compare the performance of two anticipatory algorithms against a myopic approach by focusing on analysis with practical relevance, e.g. interaction and synchronization with the real world. Similar work has been done for problem classes by Ghiani, Manni, & Thomas (2012) for a *dynamic stochastic traveling salesman problem (DSTSP)*, and in Ulmer (2019) considering a *DVRP with stochastic requests* for one vehicle. The first algorithm for comparison is a sample-scenario approach, or a *look-ahead algorithm* as defined in Ulmer et al. (2020). Sampling approaches allow detailed short-term anticipation, which is advantageous for problems with a high degree of dynamism and applicable with large problem sizes. A promising algorithm in this category is the *stochastic variable neighborhood search (SVNS)* introduced by Gutjahr, Katzensteiner, & Reiter (2007). It is the stochastic variant of the well-known *variable neighborhood search (VNS)* introduced by Hansen, Mladenović, Todosijević, & Hanafi (2017), where possible future scenarios are sampled and used to compare two solutions. Schilde et al. (2011) and Sarasola, Doerner, Schmid, & Alba (2016) successfully implemented a dynamic SVNS, and results showed an increase in solution quality compared to a myopic approach. The second algorithm implements a waiting strategy (*policy function approximation* in Ulmer et al. (2020)). Experiments in Schilde et al. (2011) show that the best results are obtained by using only a single scenario with future requests in the near future. Based on that assumption, the pivotal question is whether to wait at a location (because it is likely that new requests will occur nearby) or go to the next location in the current plan. Therefore, we implemented two different waiting strategies: one similar to the work in Vonolfen & Affenzeller (2014), and another that fairly distributes the wait times of the routes. Other fruitful waiting strategies can be found in Mitrovic-Minic & Laporte (2004), Branke, Middendorf, Noeth, & Dessouky (2005), and Thomas (2007). To guarantee a fair comparison, the information about future events is gathered from the same stochastic model, and we used the same basic optimization algorithms (dynamic VNS).

2.6. Dial-a-ride problem

The comparison of our algorithms is performed on a transportation problem, called the *dial-a-ride problem (DARP)*, where patients and elderly people are transported. Cordeau & Laporte (2007) and Parragh, Doerner, & Hartl (2008) both give a summary of DARP models and algorithms, and Molenbruch, Braekers, & Caris (2017) and Ho et al. (2018) provide recent reviews of DARPs. The DARP is an NP-hard problem (Healy & Moll, 1995) and for the static variant many sophisticated and efficient solution approaches have been proposed, such as in Cordeau & Laporte (2003), Parragh, Doerner, & Hartl (2010), Parragh & Schmid (2013), Kirchler & Wolfler Calvo (2013), Gschwind & Irnich (2015), Ritzinger, Puchinger, & Hartl (2016a), Masmoudi, Braekers, Masmoudi, & Dammak (2017), and Gschwind & Drexl (2019). The dynamic variant of the DARP (DDARP), where some information is revealed during the day of operation, is considered in Attanasio, Cordeau, Ghiani, & Laporte (2004), Berbeglia, Cordeau, & Laporte (2010), Wong, Han, & Yuen (2014), Häll, Lundgren, & Voß (2015), Santos & Xavier (2015), and Lois & Ziliaskopoulos (2017). New customer requests are considered as the event that requires a replanning of the current plan. Only Beaudry, Laporte, Melo, & Nickel (2010) considers additional events, such as vehicle breakdowns. Another variant is the *dynamic and stochastic DARP (DS-DARP)*, where different types of stochastic information, such as future requests presented in Xiang, Chu, & Chen (2008), Schilde et al. (2011), Hyytiä, Penttinen, & Sulonen (2012), or stochastic travel times discussed in Schilde, Doerner, & Hartl (2014), are incorporated into the solution approaches. In contrast to the work in Schilde et al. (2011), further constraints, like multiple depots, lunch breaks, different transportation modes, and different vehicle types, are added to conjugate with the real-world scenario. Furthermore, here we incorporate stochastic information about all patient requests, and not only the return transport trips.

3. Problem description

Our work is motivated by a real-world application of an emergency medical service in Vienna, Austria. The organization is responsible for handling the bulk of the patient transportation requests in this city. The aim is to complete transportation requests between pickup and delivery locations under user inconvenience considerations. We thus model the problem as a DSDARP, where patients can request a transport from their home location to a medical facility (outbound request) or after the medical treatment from the facility back home (inbound request). In contrast to the literature, where time windows are given at the delivery location for outbound requests and the pickup location for inbound requests, here the time windows are always determined at the pickup location and are communicated to the patient.

Each transportation request r consists of a pair of nodes (p_r, d_r), where p_r represents the pickup location and d_r the delivery location. As we are dealing with a DDARP, there is a *call time* c_r specified for each request r. For requests known in advance the call time happens before the start of the day, thus $c_r = 0$, while for dynamic requests the call time is during the day with $c_r > 0$. A time window is assigned to each pickup location $[e_{p_r}, l_{p_r}]$ and to each delivery location $[e_{d_r}, l_{d_r}]$. It defines the earliest possible start time e_{p_r} (e_{d_r}) and latest possible start time l_{p_r} (l_{d_r}). The length of the time window at the pickup node is defined according to the priority of a request $o_r = \{low, medium, high\}$. As requests cannot be rejected, the time windows are modeled to be soft, and so we have to ensure that new requests can be inserted into any solution. Starting the service after l_{p_r} (respectively l_{d_r}) is allowed but leads to a penalization (lateness) in the objective function. Patients may demand one of three transportation modes and they may or may not be accompanied by a carer. The resulting demand of a pickup node is either seated $q^1_{p_r} = 1$, lying $q^2_{p_r} = 1$, or in a wheelchair $q^3_{p_r} = 1$, and whenever the patient is accompanied, $q^1_{p_r}$ is increased by 1. The demand at the delivery node is exactly the negative demand of the pickup node. A similar formulation for a static DARP with heterogeneous patients is presented in Parragh (2011). In our case, some of the transport requests are declared as *internal* with a demand of 0, because goods are transported instead of patients, but since the goods are sensitive medical material, time windows and ride time limits are assigned as well. Furthermore, the *service time* needed to load the patient at the pickup location is given by s_{p_r}, while the service time to unload the patient at the delivery point is given by s_{d_r}. To address user inconvenience, one constraint in the DARP handles the *user ride time*, which ensures that the patient is not in transport for longer than a specified maximum user ride time. As we consider a real-world network, the maximum user ride time for each request u_r depends on the direct travel time $\hat{t}_{p_r d_r}$ from the pickup node to the delivery node of request r. Based on the practical experience of the emergency medical service, we allow a detour of at most 30 min: $t_{p_r d_r} \leq \hat{t}_{p_r d_r} + 30$.

All the patient requests are served by a fixed fleet of heterogeneous vehicles, which are based at different depots in Vienna. Each vehicle k has a depot o_k and a time window $[\bar{e}_k, \bar{l}_k]$ assigned, representing the start and end time of its shift. A vehicle is not allowed to leave the depot before $\bar{e}_k$ and should not return back to its assigned depot o_k later than $\bar{l}_k$. Returning later to the depot causes overtime and is penalized in the same way as violating the time window constraint at customer nodes (lateness). Since the shifts of the vehicles are longer than 6 hours, a lunch break b_k must be scheduled within a given hard time window $[e_{b_k}, l_{b_k}]$. To schedule a lunch break, the vehicle must be empty. There are three types of vehicles with three different modes of transportation available: seat, stretcher, and wheelchair. The available capacities of a vehicle are specified as a vector $C_k =< C^1_k, C^2_k, C^3_k >$, with C^1_k for seated patients, C^2_k for lying patients, and C^3_k for patients in a wheelchair. The vehicle types differ in the maximum available capacities of the transportation modes, e.g., vehicle type 1 has $C_k =< 4, 1, 1 >$, vehicle type 2 has $C_k =< 4, 0, 1 >$, and vehicle type 3 has $C_k =< 4, 0, 0 >$. Note, that *upgrading conditions* are applied as in Parragh (2009). For example, vehicle type 1 can transport either a lying patient or a patient in a wheelchair at the same time. Another example for vehicle type 1 is that if no lying patient is transported, two seated persons can be placed on the stretcher instead, but if a lying patient is aboard, the maximum capacity of seated persons is reduced by two. Since we consider a multi-depot problem and a heterogeneous fleet where vehicles have different shifts and capacities assigned, we introduce a pair of vehicle nodes (v_{k_s}, v_{k_e}) for each vehicle k. Vehicle routes start at v_{k_s} and end at v_{k_e} instead of starting and ending at a single depot node. As the number of vehicles is given, the focus is on reducing travel times to increase the number of requests fulfilled on time. The aim is to construct vehicle routes such that all patient requests are served without violating the precedence constraint, user ride time, lunch break time windows, and capacity constraints. A lexicographic objective function is used, which supposes that an order is given among the various objective functions. The primary objective is to minimize the total lateness, which consists of late arrival at patient requests and vehicle overtimes over all routes. Any ties are resolved using the secondary objective which is to minimize the total travel time of the vehicles.

The planning horizon of the problem consists of one working day $P = [0 \dots T]$. As we are dealing with a dynamic DARP, we distinguish between static requests, which are known in advance with call time set to $c_r = 0$, and dynamic requests arising during the day with call time $0 < c_r < T$. The *reaction time* of a request w_r is defined as the period between the call time c_r and the latest possible start of service at the pickup location l_{p_r}. Larsen et al. (2002) define several ways to express the level of dynamism in a dynamic routing system. The simplest one is the degree of dynamism, which is the number of dynamic requests n_{imm} relative to the total number of requests n_{tot}. If there are call times and reaction times available, a more meaningful way to express the dynamics of the system is the *effective degree of dynamism accounting for reaction times*: $edod_{tw} = 1/n_{tot} \sum_1^{n_{tot}} (1 - w_i/T)$, as also defined in Larsen et al. (2002). The service times used in the solution approaches are average values based on historical data. Often, in reality, these times cannot be met because patients are not always ready for pickup when the vehicle arrives, or the delivery at the medical facility takes longer due to unexpected events. Thus, additional dynamic events can arise whenever the service time in real life deviates from the one in the plan. In this work, we consider two variants of dynamic scenarios: (i) only new patient requests are considered as dynamic, and (ii) in addition to the dynamic requests, the service times are modeled as dynamic events.

4. Solution methods

A major aim of this paper is to compare the performance of anticipatory algorithms for a real-world patient transportation problem. To ensure that the differences in solution quality are based on how information about future events is incorporated, we implement a meta-heuristic approach that can efficiently solve the real-world DARP. The algorithm is VNS-based, using the same operators and settings for all solution approaches to guarantee a fair comparison of the performance of the anticipatory algorithms. Next, we propose a digital twin-based framework that mirrors real-world behavior. This is essential to appropriately analyze the developed algorithms in terms of solution quality and performance for an application in the field. The next section gives a detailed description of the anticipatory algorithms. First, we present a sample scenario approach (SVNS), and then the anticipatory algorithms implementing waiting strategies.

4.1. Solution approach for a real-world

DARP The base algorithm in this work is responsible for constructing a route plan for the medical emergency service fleet to serve all patient transportation requests. Many constraints must be fulfilled to cope with the real-world DARP, as specified in Section 3. Almost all constraints are covered in the literature, but the challenge is to respect all of them simultaneously. In the following, we present the most successful setup, which is the result of a comprehensive preliminary testing phase.

4.1.1. Route evaluation

The performance of heuristics in this field (in the field of rich VRPs such as multi-attribute VRPs) is tied to their ability to evaluate solutions efficiently. Route evaluation is based on sequences, where information about route segments is stored and reused in the search process. The concept of an efficient concatenation of sequences of routes is introduced in Vidal, Crainic, Gendreau, & Prins (2014). Here, segments of routes always start with a vehicle node, and forward concatenations of single nodes are performed. In order to improve computational efficiency, only the data required for the forward concatenation, like data used for feasibility checks and for computing the objective function, is stored. This means that the data holds information about the route segment and its state in the last node, but not the data for every single node on the route. For example, the accumulated violation of time windows is stored, but not the exact arrival and departure time at each node. This information does not influence the objective value, because here the user ride time is a constraint and not part of the objective function. Therefore, the calculation of the arrival and departure times at a node is performed at the end for the best solution, as proposed in Tang, Kong, Lau, & Ip (2010). The relevant data is computed by an appropriate calculator, which calculates the values for capacity, time window violations, user ride times, and the best slot for the lunch break. Below we briefly describe the components of the data calculator.

Capacity Demand per transportation modes is given for each node (pickup and delivery) as well as the capacity for the vehicle nodes. The data stored for a sequence is an array representing the currently available capacity of each transportation mode in the last node. To handle the upgrading conditions correctly, it is necessary to store a flag, i.e. whether or not there is a lying patient on board. For example, if the available capacity of a sequence is $C_k =< 0, 0, 1 >$ and forward extension at a delivery node with two seated patients is done, no information is given on whether $C_k^2 = 0$, because a lying patient or seated persons are placed at the stretcher. In cases where a lying patient is aboard, the newly available capacity after delivering two seated patients is $C_k =< 2, 0, 1 >$, but if no lying patient is loaded, the new capacity is $C_k =< 2, 1, 1 >$, because of the upgrading conditions.

Time windows The calculation of time window feasibility and violation is based on the concept of time window relaxation introduced in Nagata, Bräysy, & Dullaert (2010) and extended by Schneider, Sand, & Stenger (2013) for the VRP with time windows. The idea is that the time window penalty of a sequence is accumulated in order to meet the time window constraint. In this way, the violation of a time window is not propagated to a later node in the sequence, but the value is stored and can be used for the forward concatenation, i.e. calculation of the objective value (lateness). Since we introduce vehicle nodes, this concept works for the overtime calculation as well. The data set stored for a sequence is the accumulated time window violation, the earliest and latest possible departure time of the first node (vehicle node), and the duration of the segment.

User ride time The most challenging part is to efficiently perform calculations for the user ride time constraint. Routes and schedules are usually generated by following the principle of starting the service at a node as early as possible. Unfortunately, this principle does not work for the DARP. It can happen that starting the service later at a pickup node p_i will prevent waiting at a later node, thus decreasing the user ride time of request i (making it feasible). The crucial part in the forward concatenation is to calculate and store information about the latest possible delivery time for the *open* requests on the route. A request is called *open* if the pickup location is already visited, but not the corresponding delivery location. The implementation in our calculator for the user ride time constraint follows the work presented in Gschwind & Irnich (2015), and Gschwind & Drexl (2019).

Lunch breaks Since it is requested to schedule lunch breaks for the vehicles, a route evaluation operator is introduced to account for the best choice of time and place, based on the work in Vidal et al. (2014). The idea is that data with a scheduled lunch break and data without a scheduled lunch break are stored for every route segment. For the data set with a scheduled lunch break, the time of the lunch break is reflected in the duration of the time window calculator, and the position of the lunch break is also stored. With any forward concatenation of a node, again data with or without a scheduled lunch break is computed, and the better solution is stored for the new segment (dominance check). The best position (place and time) for the lunch break on a route is thus computed.

4.1.2. Meta-heuristic approach

Several publications show the success of VNS-based solution approaches for the DARP, as in Parragh et al. (2010), Schilde et al. (2011) and Molenbruch et al. (2017). We therefore opt for a *General VNS (GVNS)* to solve the given real-world DARP. A detailed description of the GVNS is given in Hansen et al. (2017). It consists of a *shaking* procedure (to escape the local minima) and an improvement procedure (*local search*) which is a *variable neighborhood descent (VND)*.

Initial solution A straightforward *best-fit parallel* insertion heuristic is implemented to generate an initial solution. In a first step, empty routes are constructed for each vehicle k, consisting of the associated pair of vehicle nodes (v_{k_s}, v_{k_e}). This is a reasonable approach as the size of the fleet is given and the minimization of vehicles is not an objective. The requests for insertion are sorted according to the start of the time window of the pickup nodes first, and then according to the length of the time window. The requests are then inserted iteratively into the solution. The heuristic checks all feasible insertion positions of the pickup and delivery node and selects the one with the lowest costs in terms of the objective value. This procedure is also applied to insert new patient transportation requests into the current plan. For the insertion of a new request, a *first-fit* insertion heuristic was tested as well as a *2-regret* insertion heuristic as described in Parragh & Schmid (2013) but they did not prove competitive with the best-fit insertion approach.

Local search The improvement procedure within the VNS is a VND. The VND explores several neighborhood structures in a sequential way in order to improve the current solution. We apply the sequential neighborhood change step, as described in Hansen et al. (2017), where the order of the neighborhood structures is defined. Whenever an improvement is found in some neighborhood structure, the search continues with the first neighborhood structure, otherwise the search goes on with the next neighborhood. The applied search strategy is a *first improvement* policy, and the moves in a neighborhood are explored in random order. The procedure stops when a local optimum is reached in every single neighborhood. Since a large search space must be explored (large instances) and computational times are restricted, we also introduced a move limit within a neighborhood structure. Four moves defining the neighborhood structures are applied. Whenever a move is performed, it is ensured that both the pickup and delivery node are moved. The *relocate* move shifts a request from one route to a different route, while the *exchange* move swaps the position of two pickup nodes of different routes and inserts the corresponding delivery node to the other route. Based on the *zero split* concept presented in Parragh et al. (2010), we introduced two more moves. A zero split point of a route is given whenever the vehicle load is 0, and so the *block relocate* move shifts a sub-sequence

of nodes between two zero split points from one route to another route. The *block exchange* move swaps sub-sequences of nodes between zero split points of two different routes.

General VNS The aim of the shaking procedure in the VNS is to try to escape the local minima by performing larger moves. In this work, several shaking neighborhoods are defined that differ in the number of routes, and the neighborhoods are changed cyclically. The shaking move randomly selects a given number of routes and removes a request from each route, also selected randomly. The removed requests are then re-inserted into the solution. Since only feasible solutions are created, any shake move is accepted, even if the objective is worse. After an iteration of the VNS (shaking and local search), the candidate solution will be accepted if the objective value is better than the current solution. The stopping criterion of the VNS is a time limit. Since response times are critical in real-world applications, it is ensured that the VNS stops after the given time limit. Therefore, the time limit is also propagated to the VND algorithm, because otherwise finishing the VND would cause a time limit violation.

4.2. Dynamic solution framework

The dynamic behavior of the problem requires a simulation and decision-making framework in order to effectively handle problem-specific information and enable adequate testing of the proposed algorithms. Fig. 1 depicts the framework, which consists of two major parts. One part is the digital twin, responsible for the reproduction of the real-world environment. Besides the simulation of the real world (moving the vehicles accordingly), it is also responsible for time management and for preparing the instance data (vehicle data, request information, call times, travel times, etc.). The other part is the *decision-maker*, which includes the event handling and the optimization of the problem instance obtained from the digital twin module. Based on the solution of the optimization algorithms, decisions are made and communicated back to the digital twin. In addition, this information is also handed to the background optimization, which is part of the decision-maker module, with the aim of utilizing the time between events to improve solution quality. The design of the communication between the digital twin and the decision-maker is an essential part. The goal is to define a setup, which corresponds to the given real-world problem, to ensure appropriate information about the solution quality of the algorithms, and to allow easy transition from digital twin to real-world practice. Note that many simulation frameworks in the literature focus on testing the developed dynamic algorithms but fail to address interaction with the simulation, and thus the real-world setting. Though some approaches react to new requests and integrate them into the current solution, the point in time when the solution is propagated back to the real-world is not defined. Other approaches define allowed response times (e.g. one minute), but fail to define what happens if new events arise in the meantime. Our framework addresses these points and provides a communication strategy and synchronization process between the real world and the decision-maker. At the end of the simulation (e.g. one working day), the digital twin has stored all movements of the vehicles, such as time of arrival, start of service, end of service, and the departure time of all stops. This final state is taken into account for analysis and evaluation of the performance of the algorithm. In the following, we describe the essential parts of the digital twin, while the decision-maker module consists mainly of the optimization algorithms proposed in this work.

Event handling Because quick reactions to changes in the environment are expected, the framework is set up in an event-driven way. Decisions can thereby be provided immediately after the events responsible for keeping daily business running. There are two types of events in the system. First, whenever new requests materialize, a new event is created. The second type concerns the state of the vehicles. The digital twin-based module implements a state machine where all possible vehicle states and transitions are defined. Whenever a state transition occurs (e.g. from *vehicle is en-route* to *vehicle arrives at location*), an event is raised. Note that an event is raised at all defined state transitions, but the decision on which event triggers a reoptimization of the route plan falls under the responsibility of the decision-maker. This results in different reoptimization strategies. In the default setting (strategy R^F), the decision-maker listens to *new request* events and events when a *vehicle finishes the service at any location*. However, as demonstrated in the results section, it is worth testing how the algorithms perform for various event handling mechanisms, e.g. events trigger a reoptimization when a new request arises and at *zero split points* instead of each stop location (strategy R^Z), or reoptimization is done only at *new request* events (strategy R^N). Depending on event type, a response time limit can be defined in the framework. In our case, response times are rather short (1 s) for all events. Since the driver only knows the current transportation request, it is vital that when a vehicle finishes its service at a *zero split* location, the information about the next trip gets communicated immediately. Also, if the new request requires immediate service, the new plan has to be quickly available. Response times are thus assumed to be short for all events (1 s) in this framework.

Interaction There are two directions of communications in the framework. On one hand, the digital twin module is responsible for propagating system changes and providing the current system state to the decision-maker. On the other hand, the decision-maker is in charge of sending an updated plan back, in order to continue appropriately. Thus, the transformation from real-world status to an optimization problem, as well as the transformation from a route plan back to the real world must both be provided correctly and in a well-defined manner. In this system, the interaction is event-driven. Once an event happens, the current system state is translated to a problem instance for optimization. Since the redirection of vehicles which have transport in progress is not allowed, reoptimization starts at the next *zero split point* of the route (vehicle is empty again), and currently served stops are not considered for reoptimization (frozen stops). Hence, the instances consist of all known requests (requests scheduled after the *zero split point* and if so new requests) and updated vehicle information. The start location and the start time of the vehicle are set to the values of the last frozen stop (location and end of service time). This instance is then optimized until the given time limit is reached, and the new route plan (combined with the frozen stops) is handed back to the digital twin module in order to continue. The correct transformation is a crucial part when implementing the framework, because otherwise the wrong system state is optimized, or else a failure in the route plan causes incorrect moving of the vehicles in the simulation.

Background optimization Another feature of our framework is that background optimization runs continuously as the response times are rather short. The advantage is that the time between events is used for further optimization, but with the issue that the result of the background optimization must be synchronized with the current system state. The same optimization algorithm is used in the background optimization as in the decision-maker module, but it runs in a distinct thread without a time limit. By the time the route plan is handed to the digital twin module, the background optimization is started with the same plan as the starting solution. Whenever a new event happens, the background optimization is stopped and its best known solution is synchronized with the current system state. This is important, because the state and the solution can diverge since the background optimization does not know what is going on in the simulation, and thus in the real world. A small example in Fig. 2 illustrates the importance

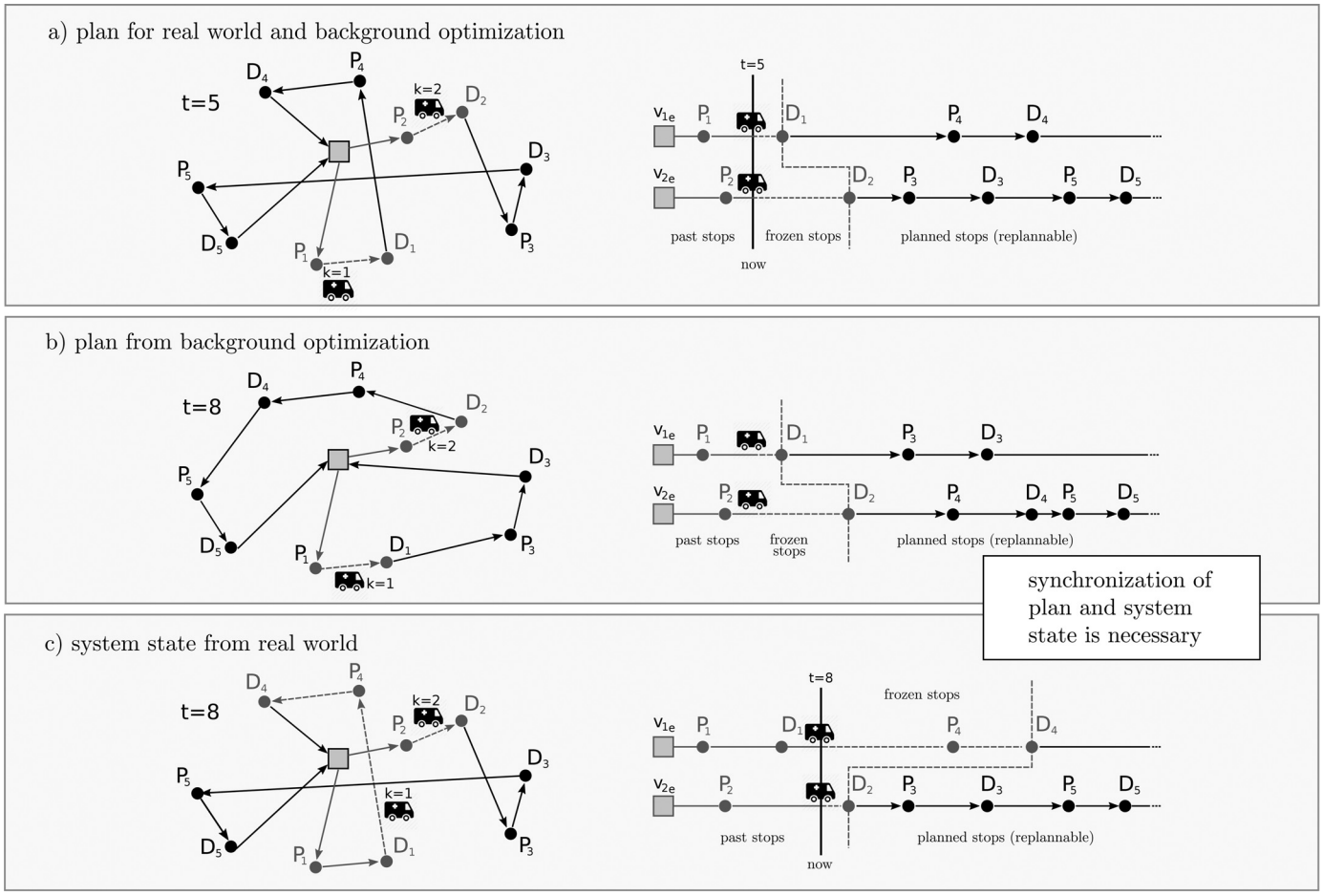


Fig. 2. This figure demonstrates the necessity of synchronization between the system state from the digital twin module (to real-world) and the plan from the background optimization for the reoptimization strategy R^N. Part a) shows a section of the plan after finishing an event handling process (e.g. new requests were inserted) at timestamp t=5. This plan is referred back to the digital twin module and the background optimization thread as well. Parts b) and c) depict the status when the next event happens at timestamp t=8. The background optimization is stopped whenever a new event happens and the current best plan is available for further optimization and depicted in part b). According to the background optimization, it is better to exchange the two requests (P_3, D_3 and P_4, D_4). Conversely, the system state in part c) shows that vehicle 1 has already departed towards P_4 and thus, cannot be replanned (frozen stop). This small example demonstrates the need for synchronization when dealing with real-world problems, which has often been neglected in the literature.

of the synchronization step. Thus, before using the plan from the background optimization in the next optimization step (considering the new event), it is adjusted to the current system state. If infeasibilities arise, the corresponding requests are removed from the solution and re-inserted in the next optimization step (together with possible new requests). After the optimization, the background optimization thread is restarted with the new plan as starting solution and it runs until another event triggers a reoptimization. In this application, inter-arrival times of events are short (due to the large instance size), which means that the time for the background optimization is short as well. Although longer runtimes are usually beneficial, it would come with the drawback of increasing the divergence between the current state and the plan from the background optimization.

4.3. Anticipatory algorithms

Based on the fact that some of the patient requests recur on a regular basis (e.g. dialysis patients) and that historical data is available (like time windows, request types, locations of pickup and delivery), information about possible future events can be exploited during the planning phase. Therefore, a stochastic model is implemented, which computes information about the expected future requests. The stochastic model aggregates geographical information and time information, computing the number of expected requests for each pair of districts within the next hour. To provide a fair comparison of the anticipatory algorithms, the information from the stochastic model is applied once as scenarios for a sample scenario planning approach, and once as demand information within a waiting strategy. The stochastic model and the anticipatory algorithms are both described below in detail.

4.3.1. Stochastic model

Demand for patient transport trips between districts is modeled for one-hour intervals using Poisson models. These generalized linear models explain the logarithm of the mean of the counts of patient transport trips μ using a linear function of explanatory variables x as $ln(\mu) = x'\beta$, where β are parameters of the model fitted to the data (see Zeileis, Kleiber, & Jackman, 2008 for details on count model regression). Similarly to Attanasio et al. (2007), the explanatory variables are time of the day in one-hour intervals ($time_{hh}$) and weekdays ($Mon - Sun$). In addition, a linear trend is added to the model as well as sine and cosine variables

$$sin\_i(t) = sin(\frac{2\pi it}{24 \cdot 365}),$$

that model yearly seasonality, where t is the time interval and $i = 1 \ldots 4$. Finally, auto-regressive variables AR_i and counts for trips in the opposite direction AR_oppi for $i = 1 \ldots 24$ are added to test

Table 1
The Poisson model for patient transport trips from the 19th district to the 9th district.

Poisson model							
Variable name	Value	Std (p-value)	Variable name	Value	Std (p-value)	Model statistics	Value
(Intercept)	−9.83***	1.02 (0)	*Mon*	1.42***	0.23 (0)	AIC	2800.16
time_7	5.69***	1.02 (0)	*Tue*	1.64***	0.22 (0)	Log likelihood	−1371.08
time_8	7.54***	1.00 (0)	*Wed*	1.51***	0.23 (0)	Deviance	1804.28
time_9	7.49***	1.00 (0)	*Thu*	1.62***	0.22 (0)	# Observations	10648.00
time_10	6.02***	1.01 (0)	*Fri*	0.88***	0.24 (0.0003)		
time_11	5.64***	1.02 (0)	*Sun*	−2.48***	0.74 (0.0007)		
time_12	5.18***	1.01 (0)	*cos_1*	0.28***	0.07 (0)		
time_13	5.44***	1.02 (0)	*cos_3*	−0.10	0.06 (0.1086)		
time_14	5.18***	1.04 (0)	*cos_4*	−0.35***	0.07 (0)		
time_15	4.44***	1.09 (0)	*sin_4*	−0.21***	0.06 (0.0007)		
time_16	4.84***	1.05 (0)	*AR_opp12*	0.61***	0.15 (0)		
time_17	3.84***	1.12 (0.0006)	*AR_opp21*	−0.16	0.10 (0.1325)		
time_18	4.13***	1.10 (0)	*AR_opp23*	0.26**	0.10 (0.0077)		
time_19	3.85***	1.12 (0.0006)	*AR_6*	−0.61	0.34 (0.0764)		
time_20	3.17**	1.23 (0.0098)					

***$p < 0.001$, **$p < 0.01$, *$p < 0.05$

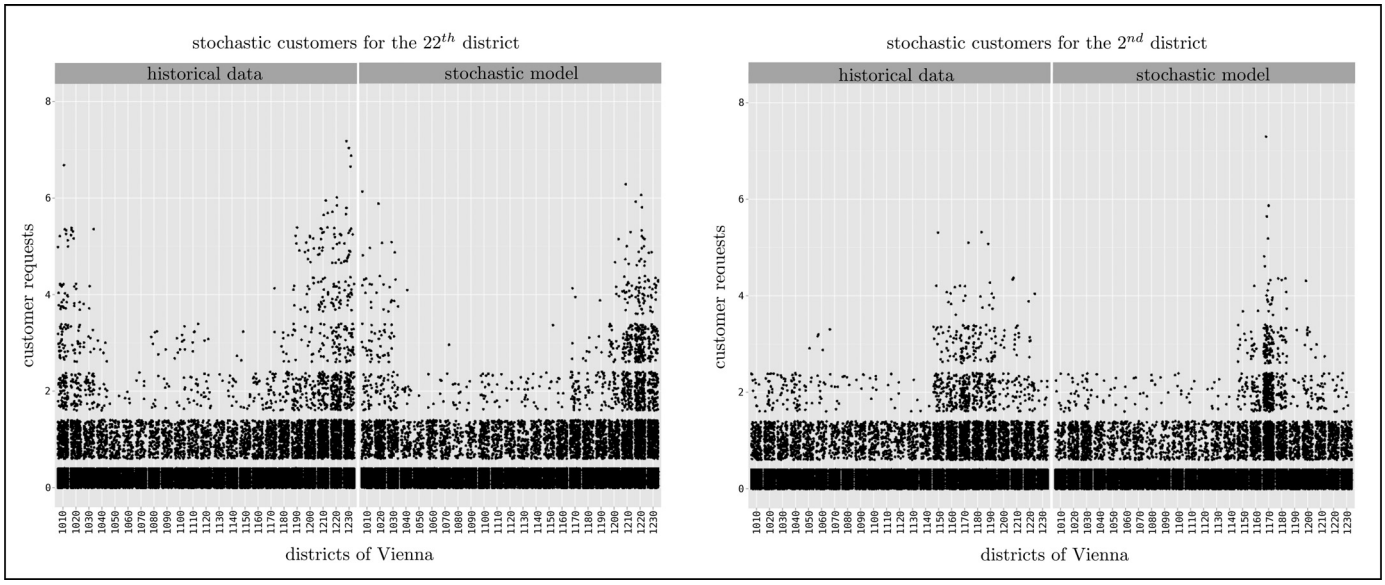


Fig. 3. This figure depicts two examples where historical data is compared against the output of the stochastic model. The number of customer requests (y-axis) from one district to all other districts in Vienna is shown for different starting times (x-axis). The left panel shows the data for the 22nd district (a suburban district), while the right panel gives another example for the 12th district (inner city district).

if return trips are booked and if trips in the same direction in previous hours somehow have an influence on the demand. To avoid overfitting of the models, a step wise algorithm is applied that takes away or adds one variable in each step such that the *Aikake Information Criterion (AIC)* is optimized (see, for instance, An & Gu (1989)). An example for a resulting model for patient transport trips from 19th district to the 9th district (where one of the main hospitals in Vienna is based) is shown in Table 1. The table shows the variables used in the model (*Variable name*), the estimated parameter values (*Value*), the standard deviation of the parameter values (*Std*), and the p-value of the student t-test for significance of the parameters (p-*value*). Looking at the parameters, we can see that the number of trips taking place is the highest from Monday to Thursday in the morning hours, and that trips in the opposite direction 12 and 23 hours before the time interval increase the number of trips in the current interval. Fig. 3 depicts the output of the model, where the instance data is compared against the output data of the stochastic model.

4.3.2. Sample-scenario planning

A sample-scenario planning approach samples stochastic information and constructs solutions based on sampled scenarios. Thus, for stochastic requests, a sample consists of one or more scenarios, i.e. possible future transportation requests. At each decision point, a static and deterministic problem is solved including the sampled information (requests) as known information. In our case, the sample is generated based on the information provided by a stochastic model.

Sampling A sampling procedure draws a sample Z of s independent scenarios $\omega_1, \ldots, \omega_s$. Each scenario consists of several possible requests, based on the output of our stochastic model. Based on the number of possible requests for each pair of districts on an hourly basis, stochastic requests are generated and added to the scenario. The call time of the stochastic request is chosen randomly within the corresponding hour, the pickup and delivery locations are chosen randomly within the corresponding district, and the time window is set randomly as well. The scenarios consist of requests for the entire instance, but which stochastic requests get used in the optimization algorithm depends on the sampling horizon S^m. At each decision point, only stochastic requests where the pickup time window starts not more than S^m minutes in the future are considered. Thus, the size of the applied scenario strongly depends on this parameter.

Stochastic VNS The implemented GVNS is extended by the SVNS concept presented in Gutjahr et al. (2007). The SVNS is based on the general structure of a VNS but differs mainly in the comparison of solutions. Every time a comparison of solutions needs to be done, a sample of s scenarios is generated as described above. All possible future requests of a scenario are inserted into the solution (best fit insertion), and the objective value is computed for the solution incorporating the future requests. In the SVNS algorithm, the solutions are compared based on the *sample average estimator (SAE)*, which is the average objective value over all s scenarios. In this way, the solution with the better average objective value with respect to possible future transports, is preferred. The implemented SVNS is similar to the original SVNS, except that the proposed *tournament step* is omitted because it extends runtime without meaningfully improving the results. The GVNS is adapted such that before every local search, a sample Z is generated that respects the number of scenarios s and the given sampling horizon S^m. The SAE is used to explore the neighborhood structure.

Every move evaluation within a neighborhood thus incorporates information about possible future requests. After the local search, a new sample is generated and the decision of whether the solution is accepted or not is again based on the SAE. Other work, e.g. Schilde et al. (2011) and Sarasola et al. (2016), omit the SAE calculation in the local search procedure for better results, but this is counter to the results of our preliminary experiments. Note that the SAE calculation takes far more runtime but enables fewer moves and iterations due to the restricted response time.

4.3.3. Waiting strategies

Results in previous work (e.g. Schilde et al., 2011) show that the most effective setting for the SVNS is to apply a single scenario and a relatively small sampling horizon. We assume this setup shows a similar behavior to a waiting strategy, since the sample consists of stochastic return transports from medical facilities in the very near future. Thus, by evaluating the solution with the sample, gaps for possible return transports are provided at the beginning of the routes of vehicles currently located at medical facilities. Based on this assumption, and the potential of waiting strategies described in previously published research, e.g. for dynamic VRPs in Ichoua, Gendreau, & Potvin (2006) and for *dynamic pickup and delivery problems (DPDP)* in Vonolfen & Affenzeller (2014), we developed a waiting strategy for the problem at hand as a counterpart to the SVNS. Another argument for applying a waiting strategy is the limited response times in our environment. Thus, fast algorithms exploiting information about future events are desirable and enable us to investigate whether such an approach can compete with a sophisticated SVNS approach. Here we implemented two different waiting strategies. The first, a *coverage algorithm (CA)*, is based on the demand information provided by the stochastic model to allow a fair comparison against the SVNS. The second approach, an *intensity algorithm (IA)*, is based on the work of Vonolfen & Affenzeller (2014) by simply utilizing historical data instead of a stochastic model. The two approaches are described below in more detail.

Coverage algorithm (CA) If wait times appear in the route plan between two locations, the algorithm aims to locate vehicles at locations in districts, where future requests are more likely to arise according to the output of the stochastic model. Note that the entire waiting time between two locations is spent either at the location after finishing the service or at the next location before starting the service. The aim of the presented CA here is to balance the number of stops with potential wait times and the number of possible future demands within each district. In contrast to a simple waiting strategy (e.g. *wait first*, as implemented in the myopic approach), the CA determines the waiting locations by an exact algorithm, running as a post-processing step every time the GVNS is finished. Therefore, an exact model is implemented and solved by an exact solver (CPLEX). To ensure the limited response time in our system, the time limit for the GVNS is reduced by the runtime of the CA (50 ms).

For the exact CA, a set of zones Z is given that consists of all districts. Additionally, a set of nodes N is defined that contains nodes with potential wait times. This means that only nodes of the route plan with node-to-node wait times are considered, the other nodes of the route plan are not included in the problem space. Set A consists of those arcs $(i, j) \in A$ with $i, j \in N$ of the route plan, where the vehicle has to wait either at node i or node j. For each zone $z \in Z$ a demand d^z is given that represents the number of possible future requests in zone z according to the stochastic model. The model consists of the decision variable $x_i^z, i \in N, z \in Z$, which is set to 1 if the vehicle has to wait at node i, and 0 otherwise. The aim is to minimize the gap between the potential wait locations x_i^z and the demand d^z for all zones. The model is defined as follows:

$$\min \quad (d^z - \sum_{i \in N} x_i^z)^2, \qquad \forall z \in Z \tag{1}$$

$$\text{subject to} \qquad x_i^z + x_j^{z'} = 1, \qquad \forall (i, j) \in A, \forall z, z' \in Z, z \neq z' \tag{2}$$

$$x_i^z \in \{0, 1\}, \qquad \forall (i) \in N, \forall z \in Z \tag{3}$$

where (1) is the objective function, minimizing the gap between wait locations and demand for all zones, and (2) ensures that the vehicle waits exactly at one node (either at node i or at node j) of the corresponding arc $(i, j) \in A$. Fig. 4 depicts the idea of the CA.

Intensity algorithm (IA) The most decisive factor for implementing another waiting strategy is the rationale for using an intensity measure instead of a stochastic model as stated in Vonolfen & Affenzeller (2014). A stochastic model needs a certain level of data quality and generally requires intensive preprocessing steps (Ferrucci, Bock, & Gendreau, 2013), but this is not always given, especially when dealing with real-world problems. Hence, the IA is a counterpart to the proposed SVNS and CA that does not require the implementation of a stochastic model. Furthermore, the IA differs from the CA in the way that the wait time is prorated between locations according to their intensity values (the CA decides to wait the total time either at the current or the next location). The intensity value is calculated based on a set of historical transportation requests $\bar{r} \in \bar{R}$, where the set $\bar{N}$ consists of the pickup and delivery node of the historical requests $\bar{r}$. Based on the intensity value, the wait time is distributed between the two corresponding nodes i and j. The intensity value considers the geographical and temporal closeness of node i in comparison to node j. For the geographical closeness, a rectangle is drawn with node i as center node, and the temporal closeness is represented by an interval around the service time t_i of node i, which is the end of service time for a departure node and the start of service time for an arrival node. The calculation of the intensity value of node i considers the subset $\hat{N}_i \subset \bar{N}$, which consists of all nodes satisfying the geographical and temporal closeness. The intensity o_i of a node is calculated by $o_i = \sum_k^{|\hat{N}_i|} \hat{t}_{ik}/|\hat{N}_i|$, and then the ratio between nodes i and j is determined as $I_{ij} = 1 - (o_i/(o_i + o_j))$. Thus, the actual wait time $\hat{o_i}$ at node i is the available wait time y_{ij} multiplied by the ratio $\hat{o_i} = y_{ij} * I_{ij}$, and the remaining time is made to wait at node j with a wait time of $\hat{o_j} = y_{ij} - \hat{o_i}$. For practical applicability and in order to reduce the organizational effort, wait times less than 5 minutes are not split, and the total wait time is spent at one location (if $I_{ij} > 0.5$ at node i).

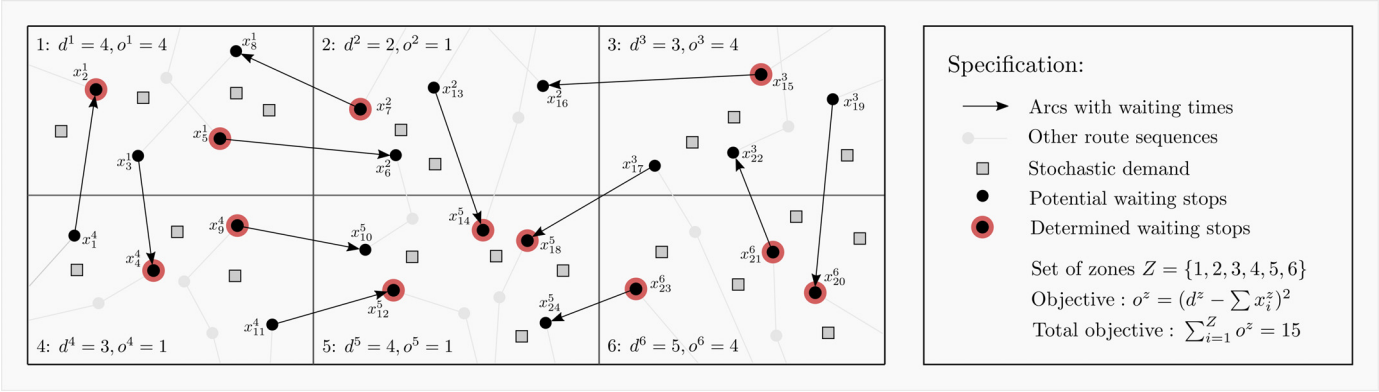


Fig. 4. A small example of the idea of the CA with 6 zones (districts), giving the stochastic demand d^z for zone z within the next hour (rectangles), and the potential waiting stops (black points) within this period. The aim is to position the vehicles such that the stochastic demand is covered in the best way, where 'covered' means to balance the difference between wait stops and demand over all districts. The objective value for each zone is specified by o^z. The red points in the figure represent the result from the CA, i.e. the determined wait stops. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5. Computational experiments

The digital twin-based framework enables an extensive evaluation of the implemented algorithms, and the computational experiments are performed on a set of test instances generated based on real-world information and data we gathered from the emergency medical service in Vienna. A detailed description of the instances is given in Section 5.1. The framework and the optimization algorithms are implemented in Java, and experimental computations are performed in the VSC-3-environment (Cluster, 2020). Theoretically, each test run lasts 18 hours, starting at the 3 a.m. timestamp by creating an initial solution of the static requests with a runtime of 3 hours. The simulation of the day of operation lasts from 6 a.m. until 9 p.m. For the sake of saving computational time, the initial solution for each instance is calculated once and used for all test runs. Thus, simulation time starts at 6 a.m. and ends at 9 p.m. The presented results are the average objective value of 20 runs. The computational effort for the results in this work is relatively high: obtaining the final results for this section comprising 7 types of algorithms, 3 reoptimization strategies, 2 types of dynamic events, 20 instances, 20 runs, and a simulation period of 15 hours results in approximately 252K hours of runtime (which includes only the final results presented in Sections 5.2.3 and 5.2.4). As stated in Schilde et al. (2011), a speed-up factor for the simulation time does not yield representative results. This is even more the case here, as the impact of background optimization is investigated. The parameters for the GVNS are based on numerous preliminary results, and in order to provide a fair comparison of the different algorithmic concepts (myopic vs. waiting strategies vs. stochastic), the same settings are used for all variants.

5.1. Instance data

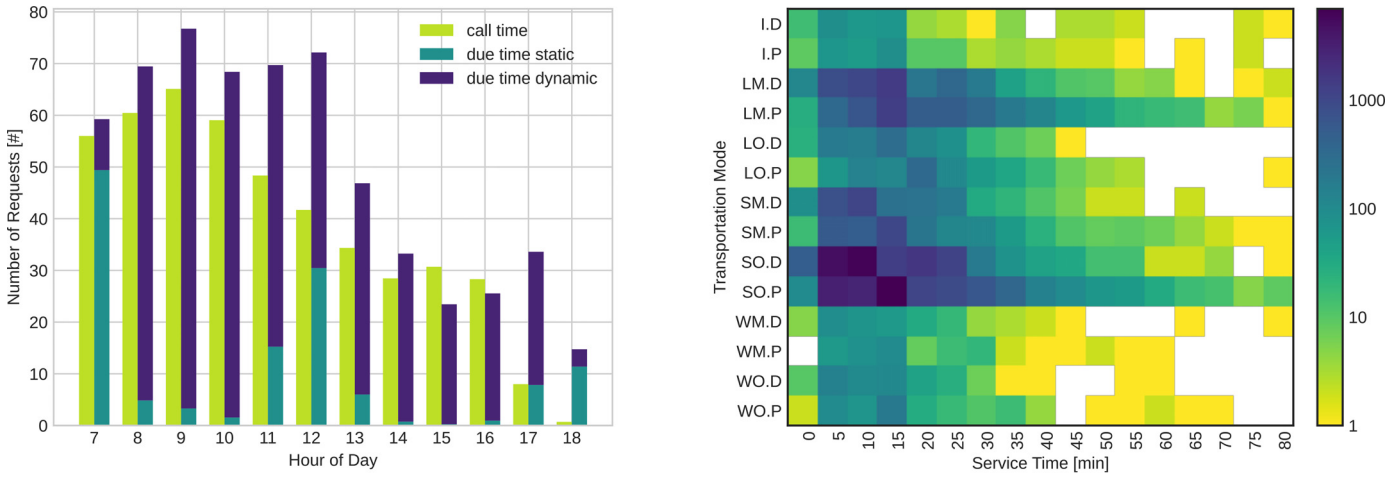
As we had available data from an emergency medical service in Vienna, the generated instances are strongly based on this data. We generate test instances to be in line with the privacy requirements of the service provider and to be able to control some of their characteristics. The data available covered daily operations for 17 months, consisting of 284,905 anonymized patient transportation requests. For the computational experiments, we generated an instance set of 4 weeks distributed over the year, considering business days (Monday to Friday), since the weekend patient transport business is sparse and not representative. This results in 20 test instances. A substantial part of the requests is dynamic and arises during the day of operation, and only a small part is available on the day before. The call time of static requests is $c_r = 0$ and the dynamic requests in the instances have their call time between 7 a.m. and 6.30 p.m., since a whole day of operation is considered. Besides the call time, each request has a given due time m_r at the pickup location and a given priority o_r. Based on these values, the soft time windows for the requests are generated such that the due time plus a buffer time according to priority specify the end of the time window at the pickup location: $l_{p_r} = m_r + o_r$, where $o_r = 10$ for low, $o_r = 5$ for medium, and $o_r = 0$ for high priority requests (in minutes). Since there is no buffer time for high-priority requests, the due date is the end of the time window $m_r = l_{p_r}$. In general, the service can start 10 minutes before the due time, thus $e_{p_r} = m_r - 10$). If a vehicle arrives before e_{p_r}, it has to wait. However, there is a part of high-priority requests that have to be served immediately (e.g. assistance with an emergency), thus requiring the start of the time window to correspond to the call time (immediate requests). For the delivery location, the time window can be restricted such that: $e_{d_r} = e_{p_r} + s_{p_r} + \hat{t}_{p_r d_r}$ and $l_{d_r} = l_{p_r} + s_{p_r} + u_r$, where s_{p_r} refers to the service time at p_r, u_r refers to the user ride time of the request, and $\hat{t}_{p_r d_r}$ is the direct travel time from pickup to delivery location.

Average number of vehicles is 49.5 (std=8.3%) and average number of transportation requests per weekday is 593.2 (std=7.1%). Table 2 gives a summary of the average statistics of the instances per week. Fig. 5 a presents the average number of call times and due times per hour for all instances. The second bar distinguishes between the number of due times from the static and the dynamic requests. The peak at 9 a.m. and 12 p.m. for the due times of static requests is based on the fact, that medical facilities have specified times for special treatments (e.g. dialysis patients). In addition, the figure shows the high dynamics of the instances, since the call times extend over the whole day.

The service time for each location in the instances is an average value of the service times in the historical data, aggregated according to pickup and delivery location and to the transportation mode of a request. The average service time at pickup locations is 16.83 minutes, while the average service time at delivery locations is 13.42 minutes. Note that these aggregated average service times are used in the optimization algorithm, while the actual service times for each location are only available in the simulator module. In this way, a set of experiments can be run where the actual service time in the real world (digital twin) may differ from the planned service times. Fig. 5 b depicts the occurrence of the actual service times of all historical requests per type (P pickup, D delivery) and transportation mode (I internal, L lying, S seated, W wheelchair, M with company, O without company). Furthermore, each pickup and delivery location is given by geographic coordi-

Table 2
Average instance data per week (Mon-Fri). Number of vehicles, total number of requests, number of static and dynamic requests, number of outbound and inbound requests, number of low and medium priority requests, number of high priority requests with the share of immediate requests, degree of dynamism (*dod*) respectively, the effective degree of dynamism ($edod_{tw}$), and average reaction time(w_r).

	vehicles	request	static	dynamic	outbound	inbound	low	medium	high	*dod*	$edod_{tw}$	w_r
$week_1$	47.2	579.4	124.0	455.4	254.6	244.4	3.8	182.4	317.6 | 75.6	0.778	0.770	26.91
$week_2$	53.2	611.0	133.8	477.2	248.4	254.0	2.4	182.8	320.6 | 105.2	0.772	0.764	27.77
$week_3$	50.4	608.0	138.0	470.0	261.8	260.6	2.2	189.0	335.2 | 81.6	0.764	0.756	29.05
$week_4$	47.0	574.2	132.2	442.0	258.0	248.8	2.2	179.6	329.2 | 63.2	0.762	0.754	27.90
avgerage	49.5	593.2	132.0	461.2	255.7	252.0	2.7	183.5	325.7 | 81.4	0.769	0.761	27.91



(a) Average call times and due times.

(b) Average service times per transportation mode.

Fig. 5. The two figures illustrate the high degree of dynamism (left) and the behavior of service times (right) of the given problem.

nates in the area of Vienna. The travel time between two locations is computed by *Ariadne*, a routing tool proposed in Prandtstetter, Straub, & Puchinger (2013). We verified in previous analyses that the travel speed of vehicles handling patient transports (in contrast to emergency calls) corresponds to the regular travel speed in Vienna. The available fleet is gathered from the historical data, and the vehicles are based at 6 different depots. Each vehicle is assigned to either a shift from 6 a.m. until 2.30 p.m. or a shift from 7 a.m. until 7 p.m., and an assigned time window within the lunch break must be held.

In addition to the instances, we also needed to generate samples for the SVNS algorithm. Each sample Z consists of s scenarios, and the number of requests for each pair of districts per hour is gathered from the stochastic model. According to this amount, possible future requests are generated such that a random pickup location and a random delivery location are selected for the appropriate districts and the due time is chosen randomly within the given hour.

5.2. Computational results

In summary, the optimization algorithms differ in the way future information is incorporated. The myopic approach (M) does not consider any future information, the two waiting strategies anticipate future information (CA and IA), and the SVNS incorporates stochastic information about future requests. SVNS variants with different settings in terms of number of scenarios s and sample horizon S^m are considered, e.g. S.10.3, with $s = 10$ and $S^m = 3$ (in minutes). Then, three different reoptimization strategies are investigated: R^F, R^Z, and R^N. All strategies perform a reoptimization when a new request arises, R^F also reoptimizes when a vehicle finishes the service at some location, and R^Z also reoptimizes when a vehicle finishes the service at a zero split location (empty vehicle). Regarding the dynamic events, two settings are investigated: first, dynamic requests (D^N), and second, dynamic requests and dynamic end times of services (D^S).

5.2.1. Analysis of the SVNS

The value of incorporating stochastic information while planning strongly depends on the sampling horizon S^m and the number of scenarios s. Based on the results in Schilde et al. (2011), we run tests with a sample horizon $S^m = 1, 3, 5, 10, 20$ minutes, and a number of scenarios $s = 1, 10, 50$. Due to the computational effort, we perform the analysis of the SVNS on two days and the best settings are then used for further experiments. In the following, we take a closer look at the performance of the SVNS, and its results are presented in Table 3. The table aggregates the results according to reoptimization strategy (*reopt*), the sampling horizon (S^m), and number of scenarios (s). The column headed *shak[#]* shows the average number of shakings in one iteration, and the column headed *saec[#]* shows the associated average number of SAE calculations. Note that for better readability, the values in the column under *saec* are in thousands, i.e. 148 means 148K. The reoptimization strategies clearly increase these numbers: for example, the number of SAE calculations is 177K with strategy R^F, but 563K with strategy R^N (for $s = 1$, $S^m = 1$). This is because the number of shakings strongly depends on the average runtime per iteration, which depends on the reoptimization strategy (see Table 4). On the other hand, these values decrease with increasing values of S^m and s, since move evaluation in the SVNS is computationally more expensive for a larger number of scenarios and a longer sampling horizon. The next column *requ* shows the average number of requests per scenario, and column *std* shows the corresponding standard deviation. These numbers are independent of the different values of s and the different reoptimization strategies, but the average number of requests per scenario increases with increas-

Table 3
Characteristics and results of different settings for the SVNS. The table shows number of shakings per iteration (*shak*), number of SAE calculations (*saec* times thousand), number of requests per scenario (*requ*), standard deviation (*std*), and lateness (*late*). All values are the average values over all instances. The last row presents the ranking of the average lateness over all reoptimization strategies and highlights the selection of four different settings (underlined and bold) for the SVNS for further experiments.

reopt	S^m	$s = 1$					$s = 10$					$s = 50$				
		shak [#]	saec [#]	requ [#]	std [%]	late [m]	shak [#]	saec [#]	requ [#]	std [%]	late [m]	shak [#]	saec [#]	requ [#]	std [%]	late [m]
R^F	1	16.8	177	0.6	4.7	1804.1	16.5	168	0.6	4.6	1714.4	14.0	148	0.6	4.5	1610.6
	3	16.5	163	1.4	6.9	1714.1	14.6	149	1.3	6.9	1694.1	12.9	136	1.3	6.9	1683.0
	5	15.0	153	2.4	9.3	1739.6	13.3	135	2.2	8.9	1735.3	12.0	122	2.2	8.9	1743.9
	10	11.7	121	4.5	12.4	1707.3	10.1	102	4.3	12.0	1652.5	9.1	90	4.3	12.0	1721.3
	20	7.8	71	7.6	12.2	1745.5	5.3	49	7.2	11.7	2016.8	4.3	40	7.2	11.6	2030.6
R^Z	1	22.7	251	0.6	4.1	1958.6	22.2	237	0.6	4.1	1779.1	19.2	213	0.5	3.9	1743.6
	3	21.5	231	1.3	6.5	1823.8	19.6	214	1.2	6.4	1862.2	17.7	194	1.3	6.4	1712.2
	5	19.7	212	2.2	8.4	1790.8	17.2	188	2.1	8.2	1712.8	15.4	168	2.1	8.2	1807.1
	10	16.2	174	4.5	12.3	1789.2	14.2	148	4.3	12.0	1680.5	12.2	129	4.3	12.0	1667.6
	20	11.1	100	7.8	12.6	1720.9	7.6	71	7.4	11.9	2035.6	5.9	59	7.4	11.9	2187.7
R^N	1	41.2	563	0.5	2.8	2489.0	38.5	534	0.4	2.6	2403.3	34.7	474	0.4	2.6	2308.7
	3	39.0	517	0.8	4.2	2400.6	37.2	490	0.8	4.0	2171.3	32.3	435	0.8	4.0	2381.0
	5	34.9	456	2.2	8.3	2251.6	29.9	407	2.1	8.0	2502.0	27.3	365	2.1	8.0	2344.9
	10	30.7	409	3.9	10.5	2352.8	24.9	338	3.7	10.3	2279.6	22.8	303	3.7	10.2	2186.6
	20	20.3	231	7.9	12.7	2346.4	14.1	162	7.6	11.9	2407.3	14.8	150	7.5	11.9	2718.4
Rank																
	S^m	1	3	5	10	20	1	3	5	10	20	1	3	5	10	20
	avg.	2083.9	1979.5	1927.3	1949.7	1937.6	1965.6	**1909.2**	1983.4	**1870.8**	2153.2	1887.7	**1925.4**	1965.3	**1858.5**	2312.2

Table 4
Benefit of applying a background optimization procedure, and effect of the different reoptimization strategies (*reopt*) for the optimization algorithms M, CA, and S.10.3. The columns headed *late* and *tt* show the average improvement (in %) with background optimization compared against periodic optimization, and the columns headed *runtime* and *shakings* show the average runtime (in seconds) and average number of shakings per iteration. The table also shows the improvement brought by the background optimization (*impr_bg*) and after the synchronization (*impr_sync*) in minutes, and the gap between the two.

algorithm	reopt	late[%]	tt[%]	events[#]	runtime[s]	shakings[#]	impr_bg[m]	impr_sync [m]	gap [m]
M	R^F	44.54	6.35	782.06	74.92	50.21	1174.43	1025.48	148.95
	R^Z	46.16	6.44	559.36	98.77	67.12	1204.13	993.20	210.94
	R^N	45.63	6.12	246.95	176.89	120.58	1513.18	970.53	542.65
CA	R^F	49.88	6.89	781.03	74.03	54.30	1109.12	973.44	135.68
	R^Z	50.22	6.88	559.10	97.75	72.96	1156.80	956.99	199.82
	R^N	49.67	6.44	246.95	175.89	134.21	1434.97	919.68	515.29
S.10.3	R^F	55.19	7.99	781.29	73.94	18.98	1136.27	1088.77	127.50
	R^Z	55.94	7.88	560.10	97.60	25.50	1173.14	984.57	188.57
	R^N	52.26	6.99	246.95	175.89	48.92	1477.65	969.65	508.00

ing sampling horizon. This is quite intuitive, but from the VNS perspective, note that with an increasing number of requests per scenario, the move evaluation takes more time, which results in fewer shakings because of the time limits. Thus, the value of incorporating stochastic information in a SVNS comes with the larger number of moves, and respectively more shakings, of other algorithms. This is an important factor in Sections 5.2.3 and 5.2.4 when the performance of the SVNS is compared to the other optimization algorithms. The last column *late* gives average lateness (primary objective) in minutes.

In order to select the settings that perform best for the SVNS, we compute the average of the primary objective values (lateness) over the different reoptimization strategies. The values are presented in the last row (*avg.*) in Table 3. Among the five settings achieving the best average values, we selected $s = 10, 50$ and $S^m = 3, 10$ to conduct further experiments.

5.2.2. Analysis of reoptimization strategies

Background optimization (continuous optimization) has the advantage of maximizing computational power utilization, but it requires a more complex and error-prone implementation and demands a well-defined synchronization between the current system state and the solution of the background optimization. This set of experiments investigates the gain of the background optimization compared to a periodic optimization (i.e. the optimization algorithm is only run when a new optimization trigger is given) and the impact of different reoptimization strategies. In Table 4, the average results over all instances are presented for the myopic approach (M), the waiting strategy (CA), and the SVNS setting (S.10.3), since the results of applying different reoptimization strategies for the other presented optimization algorithms are similar. The improvement in terms of lateness at customer locations (primary objective) is given in the column headed *late* (in %) and the improvement in travel time (secondary objective) is given in the column headed *tt* (in %). The results show that extensive implementation of the background optimization and the synchronization process is definitely worth the effort, since it achieves improvements of up to 55%. On the other hand, this is not very surprising, because the total runtime for an instance under periodic optimization is much less (i.e. the number of events times 1 second, which corresponds to the specified response time in our system) than for the background optimization (where the cumulative runtime for the background optimization is about 12 h, i.e. the period where new requests arise). The next three columns show the effect of the different reoptimization strategies. The average number of events causing a reoptimization is presented in the column headed *events*. The average runtime (in seconds) between two events, thus the runtime of background optimization, is presented in the column headed *runtime*, and the column headed *shakings* presents the average number of shakings in the VNS. It shows that the number of events for R^N (new requests cause a reoptimization) is about three times less than for R^F (new requests and the end of a service cause a reoptimization). Hence, the runtime per iteration and number of shakings increase with decreasing number of events.

This leads into the question of which are the best points to interrupt the background optimization for reoptimization. As already mentioned, the DVRP literature has largely neglected research on reoptimization strategies, but our investigation demonstrates the value of an appropriate reoptimization strategy.

To demonstrate the impact of different reoptimization strategies in combination with a background optimization, the last three columns in Table 4 compare the improvements of the primary objective (lateness). For the three algorithms (M, CA, and S.10.3), the improvement gained in the background optimization is shown under *impr_bg*, and the improvement after the synchronization procedure is shown under *impr_sync*. The last column *gap* shows the difference between *impr_bg* and *impr_sync*. Note that these values are the resulting improvements for each algorithm separately, but not the improvements over a benchmark solution. Thus, the largest improvement after the synchronization (e.g. 1088.77 in *impr_sync*) does not indicate that R^F applied with S.10.3 outperforms the other algorithms. It shows that the reoptimization strategy R^F yields the best improvement for S.10.3, compared to R^Z and R^N. It can be seen that the background optimization produces a larger early improvement, but part of that gets lost after the synchronization with the continuing real-world setting. Thus, reoptimization strategies with fewer events and longer runtimes (e.g. R^N) lead to larger improvements in the background optimization but at the same time the difference between the plan and the real world increases, which causes a greater loss of the improvement in the synchronization procedure. This comes from the fact that in a highly dynamic system, the background optimization and the real system state diverge strongly, and moves in the optimization are performed on route parts that are already ongoing in the real world. The experimental results show, that using reoptimization strategy R^F affords the largest improvements, independently of the applied optimization algorithms.

Even though the experiments and results illustrate the value of different reoptimization strategies, it should be noted that these results strongly depend on the underlying problem and its dynamic structure. For example, in less dynamic environments, or problems with smaller instances, or especially systems where longer parts of the routes are fixed (e.g. more stops until the next zero split point because of larger vehicles or because the drivers get information on next-scheduled customers that are not allowed to be replanned), it may be beneficial to apply a less frequent reoptimization strategy. Another consideration for a reoptimization strategy, i.e. the runtime for the background optimization, could be based on the diminishing returns in a heuristic search (Woodruff, Ritzinger, & Oppen, 2011), such that events triggering a reoptimization are installed according to the point when no or only minor improvements with respect to the runtime are achieved.

5.2.3. Comparison of anticipatory algorithms for D^N

Table 5 reports the results for the first variant of dynamic events, considering the occurrence of new transportation requests (D^N). The results are the average values over all instances stratified by reoptimization strategy and the different optimization algorithms. The columns headed *util* and *reqlate* deliver some insight into the results for the real-world problem at the emergency medical service, whereas the other columns focus on the objective values and the comparison of the different algorithms. The first column is average occupancy of the vehicles, which is the sum of the service times at customer locations and the sum of the travel times in relation to the shift duration. It shows that the vehicles are working nearly to full capacity (ca. 83%). The first column *reqLate* shows the percentage of patient requests with late arrival, and the next column shows the corresponding average lateness in minutes. Thus, for a bit more than a quarter of requests a late arrival occurs, and when this is the case, the average lateness is around 8 minutes. Given that we are dealing with a real-world problem and that around 13% of the requests are immediate requests, the results are satisfactory and implementable. The other columns are *traveltime*, showing the average travel time for all vehicles in minutes (secondary objective), *late* showing the average lateness in minutes, *overtime* showing the average overtime of all vehicles, and *totallate* showing the primary objective value, which is the sum of lateness plus overtime. The two columns *gap(tt)* and *gap(la)* show the average improvements in solution quality of the algorithms, as a function of travel time and total lateness compared to the myopic approach (M). The last column *gap(R)*, shows the average gap of the primary objective (total lateness) for the algorithms compared to the relevant results of the reoptimization strategy R^F. To inform discussion on the comparison of different algorithmic concepts, the rows in Table 5 are highlighted appropriately, i.e. no fill for the myopic approach (M), green for the waiting strategies (IA, CA), and yellow for the results of the SVNS. The same table layout is used later on in Table 6.

To compare the results of the anticipatory algorithms to the myopic approach for each reoptimization strategy and dynamic setting D^N, Table 5 shows that the waiting strategies achieve the largest improvement, presented in *gap(la)*. The improvement achieved by the waiting strategies is around 6%, and for R^F the best improvement is obtained by the CA, whereas for the other two reoptimization strategies, the IA works best. Regarding the solution quality of the SVNS, the best results are achieved with a sampling horizon of $S^m = 3$ minutes for all reoptimization strategies. Compared to the myopic approach, the SVNS with $S^m = 3$ registers a small improvement for all reoptimization strategies, which is not the case for the SVNS with $S^m = 10$. As already mentioned in Section 5.2.1, this can be explained by the fact that fewer shakings are performed in the SVNS compared to the other algorithms (see Table 4, column *shakings*), because the move evaluation (SAE calculations) in the SVNS is more time consuming. This may be also the reason for the negative values in *gap(tt)*. Since the objective is lexicographic, first the primary objective (lateness) is minimized, which leaves less time available for minimizing the second objective in the SVNS. Note too that the gap of the waiting strategies to the myopic approach is almost the same for all reoptimization strategies, while the gap for the SVNS deteriorates for reoptimization strategies with fewer triggers for reoptimization, e.g. R^Z. The results for the SVNS are worse in general with $S^m = 10$ and get even worse than the myopic approach with reoptimization strategy R^N. The results in the last column indicate that the reoptimization strategy R^Z yields better results than R^F because almost all algorithms (except S.50.10) obtain better results with reoptimization strategy R^Z. Furthermore, results for all algorithms are worse with reoptimization strategy R^N than with R^F. This again illustrates the impact of the different reoptimization strategies and underlines the fact that the reoptimization strategy R^N, where only new requests trigger a reoptimization, are less suitable for our problem which has a high degree of dynamism. Concluding on the results for the dynamic settings of D^N, waiting strategies yield better results than the SVNS, and the reoptimization strategy R^Z works best. More precisely, the IA with R^Z is the best setting for D^N. In terms of implementational effort, this algorithm is comparatively simple, and the incorporated information about future requests does not require a stochastic model.

5.2.4. Comparison of anticipatory algorithms for D^S

In order to evaluate the performance of the anticipatory algorithms in terms of an even higher degree of dynamism, we introduced another dynamic effect adapted from the given real-world problem: dynamic end times of services (D^S). Thus, besides the dynamic events of new requests, the service times at patient locations are also dynamic. This means that the service times in

Table 5
Average results for all optimization algorithms for the first variant of dynamic events, considering the occurrence of new requests (D^N). The differences between the reoptimization strategies and the performance of the anticipatory algorithms compared to the myopic approach are demonstrated. The left side shows characteristics of the results for the real-world application (*util*, *reqlate*), and the right side shows the comparison of the anticipatory algorithms. Working to the primary objective (total lateness), the best results are obtained by the IA with reoptimization strategy R^Z.

reopt	algorithm	util [%]	reqlate [%]	reqlate [m]	traveltime [m]	gap(tt) [%]	lateness [m]	overtime [m]	totallate [m]	gap(la) [%]	gap(R) [%]
R^F	M	83.04	29.25	8.12	6646.75		1535.11	30.28	1565.39		
	IA	82.99	28.44	7.94	6633.60	0.21	1456.80	25.37	1482.17	5.32	
	CA	82.98	28.44	7.82	6632.01	0.23	1442.39	27.04	**1469.43**	**6.13**	
	S.10.3	83.12	28.57	7.98	6677.54	-0.47	1480.88	28.65	1509.53	3.57	
	S.10.10	83.25	28.54	8.22	6716.89	-1.06	1497.41	25.98	1523.38	2.68	
	S.50.3	83.18	28.57	8.01	6691.10	-0.67	1481.62	31.04	1512.66	3.37	
	S.50.10	83.26	28.73	8.25	6726.37	-1.21	1515.42	25.75	1541.17	1.55	
R^Z	M	83.06	29.05	8.02	6651.16		1502.38	27.94	1530.32		2.24
	IA	83.03	28.30	7.78	6645.63	0.08	1413.22	26.14	*__1439.36__	**5.94**	**2.89**
	CA	83.02	28.27	7.90	6642.42	0.13	1438.56	24.88	1463.43	4.37	0.41
	S.10.3	83.18	28.35	7.90	6693.90	-0.64	1451.60	27.32	1478.92	3.36	2.03
	S.10.10	83.26	28.53	8.15	6721.23	-1.07	1482.42	25.02	1507.43	1.50	1.05
	S.50.3	83.18	28.41	7.97	6696.50	-0.69	1465.17	29.07	1494.24	2.36	1.22
	S.50.10	83.33	28.81	8.35	6744.85	-1.42	1535.07	25.56	1560.63	-1.98	-1.26
R^N	M	83.25	29.96	8.22	6714.18		1573.96	30.48	1604.43		-2.49
	IA	83.22	29.26	7.97	6708.19	0.09	1488.82	25.61	**1514.43**	**5.61**	-2.18
	CA	83.22	29.23	8.01	6705.69	0.13	1506.67	26.59	1533.25	4.44	-4.34
	S.10.3	83.39	29.69	8.15	6765.20	-0.77	1547.11	28.90	1576.01	1.77	-4.40
	S.10.10	83.53	30.26	8.74	6806.14	-1.39	1678.28	30.87	1709.14	-6.53	-12.19
	S.50.3	83.37	29.57	8.21	6757.37	-0.65	1556.01	29.68	1585.68	1.17	-4.83
	S.50.10	83.58	30.55	8.89	6821.86	-1.61	1727.19	32.13	1759.32	-9.65	-14.15

Table 6
Average results for all optimization algorithms for the second variant of dynamic events, considering the occurrence of new requests and dynamic end times of services (D^S), are presented. The differences between the reoptimization strategies and the performance of the anticipatory algorithms compared to the myopic approach are demonstrated. According to the primary objective (total lateness), the best results are obtained by the IA with the reoptimization strategy R^F.

reopt	algorithm	util [%]	reqlate [%]	reqlate [m]	traveltime [m]	gap(tt) [%]	lateness [m]	overtime [m]	totallate [m]	gap(la) [%]	gap(R) [%]
R^F	M	85.77	46.99	13.60	6665.12		4012.14	193.37	4205.51		
	IA	85.75	45.67	13.10	6668.89	-0.06	3753.74	172.64	*__3926.38__	**6.64**	
	CA	85.74	46.17	13.25	6655.91	0.14	3856.18	183.77	4039.95	3.94	
	S.10.3	85.93	46.95	13.80	6723.86	-0.87	4113.85	182.71	4296.56	-2.17	
	S.10.10	86.04	47.18	14.34	6758.49	-1.38	4281.67	188.82	4470.49	-6.30	
	S.50.3	85.93	46.85	13.86	6725.10	-0.89	4112.11	184.65	4296.76	-2.17	
	S.50.10	86.12	47.16	14.82	6782.99	-1.75	4449.79	196.99	4646.78	-10.49	
R^Z	M	85.79	47.25	13.60	6673.38		4045.27	191.94	4237.20		-0.75
	IA	85.77	46.31	13.21	6674.33	-0.02	3847.58	180.18	**4027.75**	**4.94**	-2.58
	CA	85.74	46.31	13.32	6667.21	0.09	3887.97	179.27	4067.24	4.01	-0.68
	S.10.3	85.90	46.85	13.60	6715.07	-0.62	4025.62	179.81	4205.43	0.75	2.12
	S.10.10	86.02	47.28	14.15	6754.83	-1.21	4209.98	183.06	4393.04	-3.68	1.73
	S.50.3	85.94	47.13	13.74	6727.26	-0.80	4094.01	181.90	4275.91	-0.91	0.49
	S.50.10	86.08	47.35	14.34	6775.26	-1.52	4266.88	187.52	4454.40	-5.13	4.14
R^N	M	86.05	48.82	14.24	6757.16		4342.24	195.07	4537.31		-7.89
	IA	86.04	48.20	13.91	6760.36	-0.05	4186.98	189.31	**4376.29**	**3.55**	-11.46
	CA	86.02	48.21	13.98	6755.14	0.03	4218.39	188.37	4406.76	2.88	-9.08
	S.10.3	86.20	49.01	14.33	6808.04	-0.75	4397.71	198.48	4596.19	-1.30	-6.97
	S.10.10	86.37	49.13	15.00	6865.26	-1.59	4607.48	199.33	4806.81	-5.94	-7.52
	S.50.3	86.23	48.90	14.45	6816.74	-0.88	4436.22	204.06	4640.27	-2.27	-7.99
	S.50.10	86.42	49.44	15.29	6878.60	-1.79	4707.69	209.57	4917.26	-8.37	-5.82

the real world differ from the average service times used in the optimization algorithms, and the digital twin provides these dynamic end times of services. The results of the corresponding experiments are presented in Table 6, which show that the additional dynamic element not only adversely affects solution quality and especially patient convenience but also brings disadvantages to the driver by increasing vehicle utilization and increasing overtimes. Regarding lateness at patient locations, the percentage of late arrivals increases by roughly 50% (compared to the results in Table 5), and average lateness at these locations also increases considerably to about 13 min.

The comparison of the anticipatory algorithms to the myopic approach shows that the waiting strategies yield significantly better results than the SVNS for all reoptimization strategies. For example, the IA achieves an improvement of 6.64% compared to the myopic algorithm for R^F, while the results for the SVNS are always (except for S.10.3 with R^Z) worse than the myopic approach. Similar to the results in Table 5, the SVNS with $S^m = 10$ performs worst, but in contrast to the results with D^N, where the gap in solution quality between the waiting strategies and the myopic approach is nearly the same for all reoptimization strategies, here the gap decreases with fewer triggers for reoptimization (e.g. with R^N). Another difference compared to the results with D^N concerns the reoptimization strategies, since it turns out that R^F is the best reoptimization strategy for the algorithms with waiting strategies instead of R^Z, represented by the values in the last column *gap(R)*. However, for the SVNS, reoptimization strategy R^Z still yields better results than R^F. The worst results for all optimization algorithms are obtained with the reoptimization strategy R^N, leading to the conclusion that as the degree of dynamism increases, it gets better to

apply a more frequently triggered reoptimization strategy, which ensures a smaller disagreement between the background optimization and the current system state. One explanation for the worse performance of the SVNS with D^S may be connected with the fact that the stochastic model only considers information about future requests and does not provide any stochastic information about the service times. This is true, but on the other hand, it is practically impossible to incorporate all uncertainties in a real-world application into a stochastic model. For example, in our case, there are uncertainties in travel times, vehicle break downs, and request cancellations, to name just a few. It is therefore necessary to develop algorithms that are able to respond quickly to system changes.

5.3. Summary of experiments

Our key experimental findings on the performance of anticipatory algorithms applied in a highly dynamic environment considering different variants of dynamic events and reoptimization strategies are summarized as follows:

- The extensive implementation effort of the background optimization and the synchronization process consequently required is definitely worth the effort, since it achieves average improvements of 50% on the primary objective (lateness).
- Reoptimization strategies with fewer events achieve larger improvements in the background optimization due to longer runtimes, but they then lose a significant part of these improvements in the synchronization procedure. Consequently, reoptimization strategy R^F, with the highest frequency of reoptimization triggers, yields the best results for the given highly dynamic environment, independently of the applied optimization algorithm.
- Due to the generally short and limited runtimes in our system, the SVNS yields the best results when the sample for evaluation consists of 10 scenarios with a sampling horizon of 3 minutes, independently of the reoptimization strategy.
- Our analysis of the performance of the different anticipatory algorithms finds that the waiting strategies outperform the SVNS on various points. First, the IA waiting strategy yields the best results in comparison to the myopic approach, and second, it deals best with additional uncertainties, like dynamic end times of services, without a loss in solution quality, and all at considerably less implementation effort.

6. Conclusion

This paper studied a dynamic and stochastic patient transportation problem. We provided a digital twin-based solution framework with continuous reoptimization, which enables the development and analysis of different optimization algorithms for various settings for an emergency medical facility. The big advantage of such a sophisticated framework is the possibility of investigating different scenarios and strategies in a highly dynamic environment, and it totally pays off the high implementation effort. The main objective of our work was to develop different anticipatory algorithms and to investigate which algorithm performs best according to the conditions in the real-world application such as different dynamic events, short response times, and synchronization of the solutions with the current system state. We investigated what events in the real-world should be consulted to trigger a reoptimization procedure. Different reoptimization strategies were implemented and analyzed. The results show that the frequency of reoptimization procedures is a crucial factor in such a setup, because the runtime of the background optimization depends on it. Longer runtimes led to more improvement in solution quality, but it also increased the divergence between the real-world state and the current best solution, and improvements from the background optimization were lost in the subsequent synchronization process. The reoptimization strategy where reoptimization is triggered by new requests and when a service is finished at a zero split location (R^Z) performed best when dynamic events were limited to the occurrence of new requests, but when the system had to deal with additional dynamic events, e.g. dynamic end times of services, a more frequent reoptimization strategy, which is triggered by new requests and whenever a service is finished (R^F), yielded the best results. For the comparison of anticipatory optimization algorithms, we proposed a sample scenario approach (SVNS) and two waiting strategies (CA, IA), and the results of the algorithms were compared against the myopic approach which did not incorporate any information about future requests. The rationale for comparing the performance of sophisticated algorithms, such as the SVNS, to a simpler approach like waiting strategies, was based on the settings of a SVNS approach in the literature for a related problem (Schilde et al., 2011), which indicated that a waiting strategy behaved well. This was confirmed here by extensive computational results as the waiting strategy IA, which includes stochastic information without the implementation of a stochastic model, outperformed the SVNS approach in all cases. Compared to the myopic approach, the IA achieved an improvement of almost 6% for the setup where dynamic requests were considered and a slightly higher improvement of 6.6% for the case where dynamic requests and dynamic end times of services were handled. The improvement gained by the SVNS is only 3.5% for the former case, and even worse than the myopic approach for the latter case. Furthermore, the gap between the best results of the IA and the SVNS increases, when the degree of dynamism in the system increases. Thus, based on our extensive experiments and the results produced, the SVNS is not an appropriate choice for a highly dynamic environment, which requires short and limited response times.

Advances in machine learning approaches may open opportunities for future work to focus on the application of machine learning algorithms to better learn from the historical data. Since there are so many different dynamic events in a real-world application, it could be beneficial to use machine learning algorithms to identify patterns and similar behaviors in patient requests and their service times, or in vehicle states and operations. In order to enable efficient investigation on even more dynamic and thus more realistic scenarios, our digital twin-based framework would need to be enhanced by incorporating more data models and including more information about state transitions. A well implemented, well documented, and publicly-available digital twin would also have the advantage of comparing various scenarios and even many different solution approaches efficiently and fairly, which would equally be a benefit for the research community.

Acknowledgments

This work was partially funded by the Federal Ministry of Austria for Climate Action, Environment, Energy, Mobility, Innovation and Technology (BMK) within the strategic program FITIT ModSim under grant 822739 (project HealthLog) and IV2Splus under grant 832423 (project SELECT). This financial support is gratefully acknowledged. This research work was also carried out in the framework of IRT SystemX, Paris-Saclay, France, and thus granted public funds within the scope of the French Program “æInvestissements d’Avenir”g. The authors are also very grateful for support from the Arbeiter-Samariter-Bund Österreichs, which provides us with historical data and informative insights into their daily operations. The authors also thank Gerhard Hiermann for his valuable discussions and his insightful comments. The computational results presented have been achieved using the Vienna Scientific Cluster (VSC).

References

An, H., & Gu, L. (1989). Fast stepwise procedures of selection of variables by using AIC and BIC criteria. *Acta Mathematicae Applicatae Sinica, 5*(1), 60–67. https://doi.org/10.1007/BF02006187.

Archetti, C., Feillet, D., Mor, A., & Speranza, M. G. (2020). Dynamic traveling salesman problem with stochastic release dates. *European Journal of Operational Research, 280*(3), 832–844. https://doi.org/10.1016/j.ejor.2019.07.062.

Attanasio, A., Bregman, J., Ghiani, G., & Manni, E. (2007). Real-Time fleet management at ecourier Ltd. In R. Sharda, S. Voß, V. Zeimpekis, C. D. Tarantilis, G. M. Giaglis, & I. Minis (Eds.), *Dynamic fleet management*. In *Operations Research/Computer Science Interfaces Series: 38* (pp. 219–238). Springer US. https://doi.org/10.1007/978-0-387-71722-7_10.

Attanasio, A., Cordeau, J.-F., Ghiani, G., & Laporte, G. (2004). Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. *Parallel Computing, 30*, 377–387. https://doi.org/10.1016/j.parco.2003.12.001.

Beaudry, A., Laporte, G., Melo, T., & Nickel, S. (2010). Dynamic transportation of patients in hospitals. *OR Spectrum, 32*, 77–107. https://doi.org/10.1007/s00291-008-0135-6.

Bent, R. W., & Van Hentenryck, P. (2004). Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research, 52*, 977–987. https://doi.org/10.1287/opre.1040.0124.

Berbeglia, G., Cordeau, J.-F., & Laporte, G. (2010). Dynamic pickup and delivery problems. *European Journal of Operational Research, 202*(1), 8–15. https://doi.org/10.1016/j.ejor.2009.04.024.

Branke, J., Middendorf, M., Noeth, G., & Dessouky, M. (2005). Waiting strategies for dynamic vehicle routing. *Transportation Science, 39*(3), 298–312. https://doi.org/10.1287/trsc.1040.0095.

Cimino, C., Negri, E., & Fumagalli, L. (2019). Review of digital twin applications in manufacturing. *Computers in Industry, 113*, 103130. https://doi.org/10.1016/j.compind.2019.103130.

Cluster, V.. V. S. (2020). Austrian initiative on high performance computing. http://typo3.vsc.ac.at/home Accessed: Aug 2020.

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. https://doi.org/10.1016/S0191-2615(02)00045-0.

Cordeau, J.-F., & Laporte, G. (2007). The dial-a-ride problem: Models and algorithms. *Annals of Operations Research, 153*, 29–46. https://doi.org/10.1007/s10479-007-0170-8.

Cordeau, J.-F., Laporte, G., Savelsbergh, M. W. P., & Vigo, D. (2007). Chapter 6 vehicle routing. In C. Barnhart, & G. Laporte (Eds.), *Transportation*. In *Handbooks in Operations Research and Management Science: 14* (pp. 367–428). Elsevier. https://doi.org/10.1016/S0927-0507(06)14006-2.

Crainic, T. G., Gendreau, M., & Potvin, J.-Y. (2009). Intelligent freight-transportation systems: assessment and the contribution of operations research. *Transportation Research Part C: Emerging Technologies, 17*(6), 541–557. https://doi.org/10.1016/j.trc.2008.07.002.

Ferrucci, F., Bock, S., & Gendreau, M. (2013). A pro-active real-time control approach for dynamic vehicle routing problems dealing with the delivery of urgent goods. *European Journal of Operational Research, 225*(1), 130–141. https://doi.org/10.1016/j.ejor.2012.09.016.

Gendreau, M., Guertin, F., Potvin, J.-Y., & Taillard, E. (1999). Parallel tabu search for real-time vehicle routing and dispatching. *Transportation Science, 33*, 381–390. https://doi.org/10.1287/trsc.33.4.381.

Gendreau, M., Jabali, O., & Rei, W. (2016). 50th anniversary invited article - Future research directions in stochastic vehicle routing. *Transportation Science, 50*(4), 1163–1173. https://doi.org/10.1287/trsc.2016.0709.

Ghiani, G., Guerriero, F., Laporte, G., & Musmanno, R. (2003). Real-time vehicle routing: Solution concepts, algorithms and parallel computing strategies. *European Journal of Operational Research, 151*(1), 1–11. https://doi.org/10.1016/S0377-2217(02)00915-3.

Ghiani, G., Manni, E., & Thomas, B. W. (2012). A comparison of anticipatory algorithms for the dynamic and stochastic traveling salesman problem. *Transportation Science, 46*(3), 374–387. https://doi.org/10.1287/trsc.1110.0374.

Greif, T., Stein, N., & Flath, C. M. (2020). Peeking into the void: Digital twins for construction site logistics. *Computers in Industry, 121*, 103264. https://doi.org/10.1016/j.compind.2020.103264.

Gschwind, T., & Drexl, M. (2019). Adaptive large neighborhood search with a constant-time feasibility test for the dial-a-ride problem. *Transportation Science, 53*(2), 480–491. https://doi.org/10.1287/trsc.2018.0837.

Gschwind, T., & Irnich, S. (2015). Effective handling of dynamic time windows and its application to solving the dial-a-ride problem. *Transportation Science, 49*(2), 335–354. https://doi.org/10.1287/trsc.2014.0531.

Gutjahr, W., Katzensteiner, S., & Reiter, P. (2007). A VNS algorithm for noisy problems and its application to project portfolio analysis. In J. Hromkovic, R. Královic, M. Nunkesser, & P. Widmayer (Eds.), *Stochastic algorithms: Foundations and applications*. In *Lecture Notes in Computer Science: 4665* (pp. 93–104). Springer Berlin/Heidelberg. https://doi.org/10.1007/978-3-540-74871-7_9.

Häll, C. H., Lundgren, J. T., & Voß, S. (2015). Evaluating the performance of a dial-a-ride service using simulation. *Public Transport, 7*(2), 139–157. https://doi.org/10.1007/s10479-014-1605-7.

Hansen, P., Mladenović, N., Todosijević, R., & Hanafi, S. (2017). Variable neighborhood search: Basics and variants. *EURO Journal on Computational Optimization, 5*, 423–454. https://doi.org/10.1007/s13675-016-0075-x.

Healy, P., & Moll, R. (1995). A new extension of local search applied to the dial-a-ride problem. *European Journal of Operational Research, 83*(1), 83–104. https://doi.org/10.1016/0377-2217(93)E0292-6.

Ho, S. C., Szeto, W. Y., Kuo, Y.-H., Leung, J. M. Y., Petering, M., & Tou, T. W. H. (2018). A survey of dial-a-ride problems: Literature review and recent developments. *Transportation Research Part B: Methodological, 111*, 395–421. https://doi.org/10.1016/j.trb.2018.02.001.

Hyytiä, E., Penttinen, A., & Sulonen, R. (2012). Non-myopic vehicle and route selection in dynamic DARP with travel time and workload objectives. *Computers & Operations Research, 39*(12), 3021–3030. https://doi.org/10.1016/j.cor.2012.03.002.

Ichoua, S., Gendreau, M., & Potvin, J.-Y. (2006). Exploiting knowledge about future demands for real-time vehicle dispatching. *Transportation Science, 40*(2), 211–225. https://doi.org/10.1287/trsc.1050.0114.

Jaillet, P., & Wagner, M. R. (2008). Online vehicle routing problems: A survey. In B. Golden, S. Raghavan, E. Wasil, R. Sharda, & V. Stefan (Eds.), *The vehicle routing problem: Latest advances and new challenges*. In *Operations Research/Computer Science Interfaces Series: 43* (pp. 221–237). Springer US. https://doi.org/10.1007/978-0-387-77778-8_10.

Jones, D., Snider, C., Nassehi, A., Yon, J., & Hicks, B. (2020). Characterising the digital twin: A systematic literature review. *CIRP Journal of Manufacturing Science and Technology, 29*, 36–52. https://doi.org/10.1016/j.cirpj.2020.02.002.

Kirchler, D., & Wolfler Calvo, R. (2013). A granular tabu search algorithm for the dial-a-ride problem. *Transportation Research Part B: Methodological, 56*, 120–135. https://doi.org/10.1016/j.trb.2013.07.014.

Klapp, M. A., Erera, A. L., & Toriello, A. (2018). The dynamic dispatch waves problem for same-day delivery. *European Journal of Operational Research, 271*(2), 519–534. https://doi.org/10.1016/j.ejor.2018.05.032.

Korth, B., Schwede, C., & Zajac, M. (2018). Simulation-ready digital twin for realtime management of logistics systems. In *2018 IEEE international conference on big data (big data)* (pp. 4194–4201). https://doi.org/10.1109/BigData.2018.8622160.

Laporte, G. (2007). What you should know about the vehicle routing problem. *Naval Research Logistics (NRL), 54*(8), 811–819. https://doi.org/10.1002/nav.20261.

Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science, 43*, 408–416. https://doi.org/10.1287/trsc.1090.0301.

Larsen, A., Madsen, O. B. G., & Solomon, M. M. (2002). Partially dynamic vehicle routingmodels and algorithms. *Journal of the Operational Research Society, 53*, 637–646. https://doi.org/10.1057/palgrave.jors.2601352.

Lin, C., Choy, K. L., Ho, G. T. S., Lam, H. Y., Pang, G. K. H., & Chin, K. S. (2014). A decision support system for optimizing dynamic courier routing operations. *Expert Systems with Applications, 41*(15), 6917–6933. https://doi.org/10.1016/j.eswa.2014.04.036.

Liu, M., Fang, S., Dong, H., & Xu, C. (2020). Review of digital twin about concepts, technologies, and industrial applications. *Journal of Manufacturing Systems*. https://doi.org/10.1016/j.jmsy.2020.06.017.

Lois, A., & Ziliaskopoulos, A. (2017). Online algorithm for dynamic dial a ride problem and its metrics. *Transportation Research Procedia, 24*, 377–384. https://doi.org/10.1016/j.trpro.2017.05.097. 3rd Conference on Sustainable Urban Mobility, 3rd CSUM 2016, 26 27 May 2016, Volos, Greece

Lund, K., Madsen, O. B. G., & Rygaard, J. M. (1996). Vehicle routing problems with varying degrees of dynamism. *Technical report*. IMM, The Department of Mathematical Modelling, Technical University of Denmark.

Masmoudi, M. A., Braekers, K., Masmoudi, M., & Dammak, A. (2017). A hybrid genetic algorithm for the heterogeneous dial-a-ride problem. *Computers & Operations Research, 81*, 1–13. https://doi.org/10.1016/j.cor.2016.12.008.

Mitrovic-Minic, 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. https://doi.org/10.1016/j.trb.2003.09.002.

Molenbruch, Y., Braekers, K., & Caris, A. (2017). Typology and literature review for dial-a-ride problems. *Annals of Operations Research, 259*, 295–325. https://doi.org/10.1007/s10479-017-2525-0.

Nagata, Y., Bräysy, O., & Dullaert, W. (2010). A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research, 37*(4), 724–737. https://doi.org/10.1016/j.cor.2009.06.022.

Najmi, A., Rey, D., & Rashidi, T. H. (2017). Novel dynamic formulations for real-time ride-sharing systems. *Transportation Research Part E: Logistics and Transportation Review, 108*, 122–140. https://doi.org/10.1016/j.tre.2017.10.009.

Oyola, J., Arntzen, H., & Woodruff, D. L. (2017). The stochastic vehicle routing problem, a literature review, part II: Solution methods. *EURO Journal on Transportation and Logistics, 6*, 349–388. https://doi.org/10.1007/s13676-016-0099-7.

Parragh, S., Doerner, K., & Hartl, R. (2008). A survey on pickup and delivery problems. Part II: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft, 58*, 81–117. https://doi.org/10.1007/s11301-008-0036-4.

Parragh, S. N. (2009). *Ambulance routing problems with rich constraints and multiple objectives*. University of Vienna, Department of Business Administration Phd thesis.

Parragh, S. N. (2011). Introducing heterogeneous users and vehicles into models and algorithms for the dial-a-ride problem. *Transportation Research Part C: Emerging Technologies, 19*(5), 912–930. https://doi.org/10.1016/j.trc.2010.06.002.

Parragh, S. N., Doerner, K. F., & Hartl, R. F. (2010). Variable neighborhood search for the dial-a-ride problem. *Computers & Operations Research, 37*, 1129–1138. https://doi.org/10.1016/j.cor.2009.10.003.

Parragh, S. N., & Schmid, V. (2013). Hybrid column generation and large neighborhood search for the dial-a-ride problem. *Computers & Operations Research, 40*(1), 490–497. https://doi.org/10.1016/j.cor.2012.08.004.

Pillac, V., Gendreau, M., Guéret, C., & Medaglia, A. (2013). A review of dynamic ve-

hicle routing problems. *European Journal of Operational Research, 225*(1), 1–11. https://doi.org/10.1016/j.ejor.2012.08.015.
Pillac, V., Guéret, C., & Medaglia, A. (2012a). An event-driven optimization framework for dynamic vehicle routing. *Decision Support Systems, 54*(1), 414–423. https://doi.org/10.1016/j.dss.2012.06.007.
Pillac, V., Guéret, C., & Medaglia, A. (2012b). A fast re-optimization approach for dynamic vehicle routing. In *Doctoral dissertation, ecole des mines de nantes*.
Powell, W. B. (2019). A unified framework for stochastic optimization. *European Journal of Operational Research, 275*(3), 795–821. https://doi.org/10.1016/j.ejor.2018.07.014.
Powell, W. B., Simao, H. P., & Bouzaiene-Ayari, B. (2012). Approximate dynamic programming in transportation and logistics: A unified framework. *EURO Journal on Transportation and Logistics, 1*(3), 237–284. https://doi.org/10.1007/s13676-012-0015-8.
Prandtstetter, M., Straub, M., & Puchinger, J. (2013). On the way to a multi-modal energy-efficient route. In *IECON 2013 - 39th annual conference of the ieee industrial electronics society* (pp. 4779–4784). https://doi.org/10.1109/IECON.2013.6699908.
Psaraftis, H. N. (1995). Dynamic vehicle routing: Status and prospects. *Annals of Operations Research, 61*, 143–164. https://doi.org/10.1007/BF02098286.
Psaraftis, H. N., Wen, M., & Kontovas, C. A. (2016). Dynamic vehicle routing problems: Three decades and counting. *Networks, 67*(1), 3–31. https://doi.org/10.1002/net.21628.
Ritzinger, U., Puchinger, J., & Hartl, R. F. (2016a). Dynamic programming based metaheuristics for the dial-a-ride problem. *Annals of Operations Research, 236*, 341–358. https://doi.org/10.1007/s10479-014-1605-7.
Ritzinger, U., Puchinger, J., & Hartl, R. F. (2016b). A survey on dynamic and stochastic vehicle routing problems. *International Journal of Production Research, 54*(1), 215–231. https://doi.org/10.1080/00207543.2015.1043403.
Santos, D. O., & Xavier, E. C. (2015). Taxi and ride sharing: A dynamic dial-a-ride problem with money as an incentive. *Expert Systems with Applications, 42*(19), 6728–6737. https://doi.org/10.1016/j.eswa.2015.04.060.
Sarasola, B., Doerner, K. F., Schmid, V., & Alba, E. (2016). Variable neighborhood search for the stochastic and dynamic vehicle routing problem. *Annals of Operations Research, 236*, 425–461. https://doi.org/10.1007/s10479-015-1949-7.
Savelsbergh, M., & Van Woensel, T. (2016). 50th anniversary invited articlecity logistics: Challenges and opportunities. *Transportation Science, 50*(2), 579–590. https://doi.org/10.1287/trsc.2016.0675.
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. https://doi.org/10.1016/j.cor.2011.02.006.
Schilde, M., Doerner, K. F., & Hartl, R. F. (2014). Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem. *European Journal of Operational Research, 238*(1), 18–30. https://doi.org/10.1016/j.ejor.2014.03.005.
Schneider, M., Sand, B., & Stenger, A. (2013). A note on the time travel approach for handling time windows in vehicle routing problems. *Computers & Operations Research, 40*(10), 2564–2568. https://doi.org/10.1016/j.cor.2013.02.002.
Schorpp, S. (2010). *Dynamic fleet management for international truck transportation focusing on occasional transportation Tasks*. University of Augsburg, Faculty of Economics and Business Administration Phd thesis.
Speranza, M. G. (2018). Trends in transportation and logistics. *European Journal of Operational Research, 264*(3), 830–836. https://doi.org/10.1016/j.ejor.2016.08.032.
Steever, Z., Karwan, M., & Murray, C. (2019). Dynamic courier routing for a food delivery service. *Computers & Operations Research, 107*, 173–188. https://doi.org/10.1016/j.cor.2019.03.008.
Tang, J., Kong, Y., Lau, H., & Ip, A. W. H. (2010). A note on æefficient feasibility testing for dial-a-ride problemsg. *Operations Research Letters, 38*(5), 405–407. https://doi.org/10.1016/j.orl.2010.05.002.
Thomas, B. W. (2007). Waiting strategies for anticipating service requests from known customer locations. *Transportation Science, 41*(3), 319–331. https://doi.org/10.1287/trsc.1060.0183.
(2001). In P. Toth, & D. Vigo (Eds.), *The vehicle routing problem*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.
Ulmer, M. W. (2019). Anticipation versus reactive reoptimization for dynamic vehicle routing with stochastic requests. *Networks, 73*(3), 277–291. https://doi.org/10.1002/net.21861.
Ulmer, M. W., Goodson, J. C., Mattfeld, D. C., & Thomas, B. W. (2020). On modeling stochastic dynamic vehicle routing problems. *EURO Journal on Transportation and Logistics, 9*(2), 100008. https://doi.org/10.1016/j.ejtl.2020.100008.
Ulmer, M. W., Mattfeld, D. C., & Köster, F. (2018). Budgeting time for dynamic vehicle routing with stochastic customer requests. *Transportation Science, 52*(1), 20–37. https://doi.org/10.1287/trsc.2016.0719.
Van Heeswijk, W. J. A., Mes, M. R. K., & Schutten, M. J. (2019). The delivery dispatching problem with time windows for urban consolidation centers. *Transportation Science, 53*(1), 203–221. https://doi.org/10.1287/trsc.2017.0773.
Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. (2014). A unified solution framework for multi-attribute vehicle problems. *European Journal of Operational Research, 234*(3), 658–673. https://doi.org/10.1016/j.ejor.2013.09.045.
Voccia, S. A., Campbell, A. M., & Thomas, B. W. (2019). The same-day delivery problem for online purchases. *Transportation Science, 53*(1), 167–184. https://doi.org/10.1287/trsc.2016.0732.
Vonolfen, S., & Affenzeller, M. (2014). Distribution of waiting time for dynamic pickup and delivery problems. *Annals of Operations Research, 236*, 359–382. https://doi.org/10.1007/s10479-014-1683-6.
Wong, K. I., Han, A. F., & Yuen, C. W. (2014). On dynamic demand responsive transport services with degree of dynamism. *Transportmetrica A: Transport Science, 10*(1), 55–73. https://doi.org/10.1080/18128602.2012.694491.
Woodruff, D. L., Ritzinger, U., & Oppen, J. (2011). Research note: The point of diminishing returns in heuristic search. *International Journal of Metaheuristics, 1*(3), 222–231. https://doi.org/10.1504/IJMHEUR.2011.041195.
Xiang, Z., Chu, C., & Chen, H. (2008). The study of a dynamic dial-a-ride problem under time-dependent and stochastic environments. *European Journal of Operational Research, 185*(2), 534–551. https://doi.org/10.1016/j.ejor.2007.01.007.
Yan, X., Xiao, B., Xiao, Y., Zhao, Z., Ma, L., & Wang, N. (2019). Skill vehicle routing problem with time windows considering dynamic service times and time-skill-dependent costs. *IEEE Access, 7*, 77208–77221. https://doi.org/10.1109/ACCESS.2019.2919963.
Zeileis, A., Kleiber, C., & Jackman, S. (2008). Regression models for count data in R. *Journal of Statistical Software, 27*(8), 1–25. https://doi.org/10.18637/jss.v027.i08.