universität wien

DISSERTATION

Titel der Dissertation

Solution methods for the dynamic stochastic dial-a-ride problem with time-dependent travel speeds

Verfasser

Mag. Michael Schilde

angestrebter akademischer Grad

Doctor of Philosophy (PhD)

Wien, im Juli 2011

Studienkennzahl:	A 094 146
Dissertationsgebiet lt. Studienblatt:	Management
Betreuer:	o. Univ.Prof. Dipl.-Ing. Dr. Richard F. Hartl

Acknowledgements

First, I would like to express my gratitude to o.Univ.Prof. Dr. Karl F. Doerner and o.Univ.Prof. Dipl.-Ing. Dr. Richard F. Hartl for giving me the opportunity to work on this highly interesting and challenging subject, for supervising and supporting my work and also for granting my PhD research position (Austrian Science Fund, FWF Translational Research under grant #L510-N13 and Austrian Research Promotion Agency, FFG ModSim under grant #822739). Further thanks are due to my great colleagues at the Chair for Production and Operations Management for their numerous ideas, their constructive feedback and our lively discussions during the years. Last but not least I want to thank my parents and all my friends for their indefatigable support during all the years as well as for their patience.

The computational results presented have been achieved (in part) using the Vienna Scientific Cluster (VSC).

Contents

List of Tables

List of Figures

List of Algorithms

1 Introduction

The field of research regarding the optimization of humanitarian aid as well as health care efforts is very wide. During the last decades, the efforts expended in research projects related to this area are steadily increasing. This continuously growing demand may be caused by many influence factors. The constantly increasing expectation of life in most developed countries induces an increasing demand for health care as well as for transportation services suited for elderly and disabled persons. Also, the ongoing growth of human settlements in coastal areas, earthquake zones and other endangered regions leads to continuously increasing impacts of natural disasters on human life. Sadly, ongoing armed conflicts, famines, droughts and forest fires are still forcing millions of people worldwide to seek refugee or to require humanitarian aid. This list is by far not complete and thus the need for effective and efficient means of providing humanitarian aid and health care will most likely continue to get more and more important in the future.

In this book we study one selected aspect out of the wide health care subject. We investigate possible improvements regarding the logistics involved in the transportation of elderly, disabled and ill persons. The problem we address is motivated by the patient transportation tasks performed by organizations like the Austrian Red Cross (ARC) or the Arbeiter-Samariter-Bund Österreichs (ASBÖ) in their daily operation. The ARC is a non-governmental organization and the largest ambulance service provider in Austria. It runs 141 district offices, 442 offices with 24/7 service and has 5,620 full-time employees, 4,047 civilian servants and 51,430 volunteer helpers. Its vehicle fleet consists of 1,978 vehicles performing a total of 2,767,490 operations with a mileage of 94,164,699 kilometers in the year 2009 [1]. The ASBÖ is also a non-governmental Austrian organization which provides ambulance services. It has 4,694 active members of which 1,041 are full-time employees. Its vehicle fleet consists of 508 vehicles performing almost 720,000 operations with a mileage of 17,241,806 kilometers in the year 2009 [2].

One of the operations performed by these organizations is to provide (non emergency) ambulance service. Patients can call a service line in order to arrange a transport from their respective home location to a hospital (outbound requests). This service is often used by patients who need dialysis treatment, patients having mobility problems and elderly persons. After the treatment or surgery at the hospital is completed, the patients are transported back to their home location (inbound requests), if required. The vehicle fleet used for this purpose consists mainly of patient transport ambulances (PTA) that were donated by companies. Therefore, both organizations dispose of quite a large number of vehicles, whereas the number

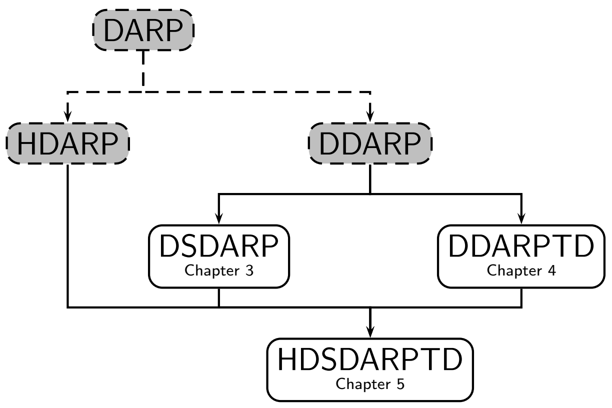

Figure 1.1: Outline of the studied problems.

of drivers available is limited. The most important aim is to cause as little inconvenience for the patients as possible. This implies that rejecting a transportation request is not an option and complying to the patient's time constraints is essential (i.e., picking the patient up and arriving at the hospital in time). Additionally, the costs caused by providing this kind of service should also be kept as low as possible.

This kind of ambulance service is offered all over Austria. In this book we will address the optimization of the vehicle dispatching operations involved in this service. Currently, this task is performed by human dispatchers. What makes this problem especially complicated is the fact that patients can arrange the transport whenever they want, which can lead to two different situations. If the transport is organized the day before it is supposed to take place (or earlier), it can be regarded as static while planning. Otherwise, the organizations currently do not have any a priori information about the request, which therefore is dynamic. Furthermore, all outbound requests (static as well as dynamic) can cause a corresponding inbound request, which is dynamic as well.

Based on a variant of the dynamic dial-a-ride problem (DDARP), which is an extension to the well known dial-a-ride problem (DARP), we study three different stochastic extensions of this problem. We adapt two pairs of metaheuristic solution approaches to each of these problems in order to compare the results of myopic solution methods to the results obtained by their stochastic counterparts. First, we study the effect of exploiting stochastic information about future outbound requests while planning the vehicle routes for the Austrian ambulance service providers. Therefore, we present the dynamic stochastic dial-a-ride problem with expected return transports (DSDARP) as an extension to the basic DDARP. Second, we exploit stochastic information about future time-dependent travel speeds while planning. For this purpose we use the dynamic dial-a-ride problem with stochastic time-dependent travel speeds (DDARPTD), which is again a stochastic variant of the basic DDARP. Third, we combine these two problem variants into one singe problem and add aspects of the heterogeneous dial-a-ride problem (HDARP). The resulting problem is called heterogeneous dynamic stochastic dial-a-ride problem with stochastic time-dependent travel speeds (HDSDARPTD). An outline of the studied problem types is given in Figure 1.1.

The remainder of this book is structured as follows. In Chapter 2 we give a short description of the DDARP which is the basis for all our stochastic problem extensions. In Chapter 3 we present our study on the DSDARP. The second stochastic extension, the DDARPTD, is described in Chapter 4. The HDSDARPTD is described in Chapter 5. The book concludes with a summary and some final conclusions in Chapter 6.

2 The dynamic dial-a-ride problem

2.1 Related work

The daily operation of ambulance service providers can be characterized as a point-to-point passenger transportation problem. Such problems are commonly referred to as dial-a-ride problem (DARP) in the literature. The DARP was introduced in the early 1970s by Wilson et al. [3, 4, 5] and has since then attracted considerable attention by the scientific research community. Healy and Moll [6] showed that the DARP is NP-hard and thus much effort has been dedicated to the development of (meta-)heuristic solution approaches for this problem class. In particular, real world motivated dial-a-ride problems have been widely studied during the last decades (see [7, 8] for recent surveys). The static and deterministic variants of the DARP are well studied and very sophisticated solution approaches were presented within the last years (see, e.g., [9, 10] for state-of-the-art metaheuristics).

In the special case faced by the Austrian Red Cross and the Arbeiter-Samariter-Bund Österreichs, the patients are free to express their need for transportation while the day progresses. This means, that some of the transportation requests are dynamic in the sense that no information about them is known a priori. Therefore, the problem at hand can in general terms be defined as a dynamic dial-a-ride problem (DDARP).

Several variants of the DDARP have lately been studied in the literature. One of the first approaches to solving the DDARP was taken by Psaraftis [11]. He proposed an exact dynamic programming approach for the static single-vehicle DARP and used it to repeatedly solve the static problem arising when a new request gets known and the already executed partial route is fixed. An insertion based algorithm for the real-life multi-vehicle DDARP was presented by Madsen et al. [12]. Horn [13] developed a heuristic algorithm for demand-responsive passenger services including time windows, capacity constraints and booking cancelations. Attanasio et al. [14] proposed a parallel tabu search algorithm for the DDARP, which in multiple parallel threads performs random insertions of a new request in an existing solution followed by a tabu search procedure for each thread. Berbeglia et al. [15] studied a hybrid tabu search and constraint programming algorithm for the DDARP. For a recent literature review on the DDARP and other dynamic pickup and delivery problems, the interested reader is referred to Berbeglia et al. [16]. A recent survey covering a larger number of different pickup and delivery problems was presented by Parragh et al. [7].

2.2 Problem definition

As a base for the following extensions (see Chapters 3, 4, and 5) we model the fundamental problem faced by the Austrian ambulance service providers as a dynamic dial-a-ride problem (DDARP). The following problem definition is based on the ones used in [17] and [18]. The problem is defined on a directed real world road network $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ consisting of a set of vertices $\mathcal{V}$ and a set of arcs $\mathcal{A}$ connecting all vertices. Each vertex $v \in \mathcal{V}$ represents a patient's home location or a hospital site. The aim of the problem is to find vehicle routes based on $\mathcal{G}$ in order to service a set of transportation requests $\mathcal{R}$ using a limited number of vehicles. Each transportation request $r \in \mathcal{R}$ is related to two vertices (p_r, d_r) representing the pickup and delivery location of the corresponding passenger, respectively. Additionally, each request r is assigned a required capacity of $q_r = 1$. All vehicles are based on a common depot location v_0 (one of the hospital locations) and is able to transport up to $Q = 3$ patients at the same time. This means that in this basic DDARP, patients and vehicles are assumed to be homogeneous in the sense that each patient occupies exactly one seat in a vehicle and each vehicle has exactly 3 seats installed.

Each request r has two time windows: $[e_p, l_p]$ for the pickup location and $[e_d, l_d]$ for the delivery location. The depot has the time window $[0, T_{max}]$.This means that all vehicles may depart from the depot from time 0 on and should return to the depot by time T_{max} at the latest. If a vehicle arrives at the depot later than T_{max}, the vehicle crew exceeds its working hours and overtime payments are to be paid (which should be avoided). All time windows are treated as soft on both ends, which means that starting service before the beginning of the time window as well as after the end of the time window is possible. However, starting service early would require the driver to wait for the patient to be ready and should therefore be avoided. Starting service after the end of the time window reduces customer satisfaction and should thus be avoided as well.

As patients are free to express their need for transportation whenever they want, some of the transportation requests are not known a priori in the DDARP. Each request r is therefore assigned a time a_r which represents the moment in time at which the request becomes known. Static requests (that were arranged before the beginning of the planning period) arise at time $a_r = 0$. Dynamic requests get known while the day progresses and thus arise at time $a_r > 0$. All outbound requests (static and dynamic) can cause a followup inbound request if the patient needs to be transported back home at a later point in time. Such return transports are arranged by the hospital personnel whenever the patient is ready to be picked up again. Therefore, such inbound requests are always dynamic as no a priori information is assumed to be known in this basic DDARP.

The classical DARP usually includes user inconvenience in terms of excess ride time as a constraint or as part of the objective function. For the problem at hand, user inconvenience is included as a maximum detour constraint in the following sense. We define the time required to travel from p_r to d_r without detour as t_{direct}.

The planned travel time $t_{planned}$ is defined as the time between the planned end of service at p_r and the planned beginning of service at d_r. The maximum detour constraint can then be stated as $t_{planned} \leq t_{direct} + 30$ in order to avoid detours of more than 30 minutes.

For the Austrian ambulance service providers, the most important aspect while planning their vehicle routes is to minimize patient dissatisfaction. Besides this, minimizing the total costs for providing the service is also important. Therefore, we define the objective function for this basic DDARP as a three-stage lexicographic function. The primary objective is to minimize the sum of earliness and tardiness as a measure for customer dissatisfaction. In order to penalize overtime payments, late returns to the depot (after the end of the depot's time window) are counted as tardiness as well. If two solutions obtain equal results regarding this primary objective, they are compared based on the secondary objective which is the number of vehicles used (i.e., the number of routes). In case two solutions are equally good in both objectives, evaluation is based on the total route duration as a tertiary objective.

3 Stochastic return transports

3.1 Introduction and related work

In Chapter 2 we present the basic version of the dynamic dial-a-ride problem faced by the Austrian ambulance service providers. However, the real-life problem includes an additional interesting aspect which we study in this chapter. This chapter is based on an article published by Schilde et al. [17] and several passages are taken from there.

Imagine a patient who needs to receive periodical medical treatment because of an illness (e.g., dialysis). Many such patients are not able to go to the hospital on their own, but need to be transported by an ambulance service provider. What makes this idea interesting for the problem at hand is the fact that the duration of such a treatment can be estimated quite precisely from historical data. Additionally, the duration of such periodic treatments is relatively independent of the person who receives the treatment or the time at which the treatment is performed. This leads us to the idea, that some stochastic information about the duration of such treatments can be derived from historical data and could as a consequence be used to improve the planning process for the Austrian ambulance service providers. Therefore, we extend the concept of the DDARP presented in Chapter 2 by introducing stochastic information about future return transports. We call the resulting problem the dynamic stochastic dial-a-ride problem with expected return transports (DSDARP).

The problem studied in this chapter is a variant of the dynamic dial-a-ride problem presented in Chapter 2 which includes stochastic aspects regarding future transportation requests. To the best of our knowledge, dial-a-ride problems with stochastic aspects related to future requests have received very little attention in the literature up to now. An article by Coslovich et al. [19] was among the first publications studying the effect of stochastic aspects regarding future transportation requests. These authors propose a two-phase insertion algorithm based on route perturbations. Xiang et al. [20] include stochastic events for new requests, the absence of customers, the cancelation of requests and more into the problem and present a heuristic solution approach for it. Hyytiä et al. [21] present a model describing a single vehicle in a dial-a-ride system with stochastic customers. Finally, Heilporn et al. [22] present an integer L-shaped algorithm for the single-vehicle DARP with stochastic customer delays.

A greater deal of attention has been attracted by other stochastic problems. Gutjahr et al. [23] developed a stochastic branch-and-bound method for optimal single-machine tardiness scheduling. Laporte and Louveaux proposed an integer L-shaped method for stochastic integer problems with complete recourse. Laporte et al. [24] proposed an integer L-shaped method for the capacitated vehicle routing problem with stochastic demands. Gutjahr [25] also proposed an ant-based approach to combinatorial optimization problems under uncertainty. Jaillet [26], Bertsimas [27], Bertsimas et al. [28] and Laporte et al. [29] studied probabilistic versions of the traveling salesman problem and other combinatorial optimization problems. Teodorović and Pavković [30], Gendreau et al. [31], and Golden and Stewart [32] published several solution methods for stochastic variants of the vehicle routing problem. Secomandi and Margot [33] proposed re-optimization approaches for the vehicle routing problem with stochastic demands. Tillmann [34] proposed a modification of the well known Clarke and Wright [35] savings algorithm for the multiple terminal delivery problem with probabilistic demands. Kleywegt et al. [36] published a method based on Markov decision processes for the stochastic inventory routing problem with direct deliveries. A solution approach to the dynamic stochastic vehicle routing problem based on a sample scenario hedging heuristic was published by Hvattum et al. [37]. Gutjahr et al. [38] recently introduced a stochastic variant of the well known variable neighborhood search (VNS, [39]) for project portfolio analysis under uncertainty. This S-VNS method is mainly based on the idea of taking possible scenarios of the future into account when comparing two solutions. This leads to very robust results. Bent and Van Hentenryck [40] proposed a multiple plan approach (MPA) and a multiple scenario approach (MSA) for the dynamic vehicle routing problem with time windows and stochastic customers. These approaches are based on the idea of maintaining a pool of solutions during execution. The MSA additionally takes into account scenarios of the future while planning. A short survey on the literature regarding stochastic vehicle routing problems was presented by Gendreau et al. [41]. Also the survey by Parragh et al. [7] contains a short overview of the literature on stochastic vehicle routing problems.

The remainder of this chapter is structured as follows. In the first section, we give a short overview on the related literature. Section 3.2 provides a short definition of the DSDARP with expected return transports based on the basic DDARP. The used solution approaches are described in Section 3.3 followed by a description of the computational experiments and results (Section 3.4) as well as a short summary (Section 3.5).

3.2 Problem definition

As described in the previous section, the problem studied in this chapter is an extension to the basic DDARP outlined in Chapter 2. This means, that the fundamental structure of the dynamic dial-a-ride problem is also valid for the dynamic stochastic dial-a-ride problem with expected return transports. The problem is also defined on a directed real world road network and aims at designing routes for a fixed fleet of homogenous vehicles located at a common depot. Each vehicle is again able to transport up to three patients at the same time and each patient occupies exactly one seat inside a vehicle. As in the case of the DDARP, all incoming transportation requests have a time window for both, the pickup and the delivery location and all requests must be serviced. The maximum detour made to pick up or drop off additional passengers is limited to 30 minutes in the same way as described for the DDARP.

The objective function of the DSDARP with expected return transports is identical to the one used for the DDARP. Again, the primary objective is to minimize the customer dissatisfaction represented by the sum of earliness and tardiness with respect to the service requests' time windows. To minimize the costs caused by a found solution, the secondary objective is to minimize the number of vehicles used. As a second factor related to the costs of a solution, the total mileage of the solutions is used as tertiary objective.

The main difference between the DDARP and the DSDARP with expected return transports is caused by a change in the underlying assumptions. In the case of the dynamic dial-a-ride problem, we assume that no a priori information of whatever kind is available about future transportation requests. For the dynamic stochastic dial-a-ride problem with expected return transports, we assume that for a part of the requests some stochastic information is available. If we take a closer look at the inbound request from a hospital location back home to the corresponding patient's home location, we see that in fact some stochastic information can be derived from historical data. Especially in the case of standard treatments like dialysis, this seems to be obvious. Such treatments can be expected to consume a more or less constant amount of time, independent of who the patient is or when the treatment takes place.

Based on this idea we assume that the time at which future inbound requests will get known can be estimated by sampling the corresponding stochastic distribution derived from historical data. An additional complexity comes from the fact that not all outbound requests cause an inbound request during the planning period. Some patients do not need to be transported back home after their treatment. They might be using a taxi instead, a relative could pick them up at the hospital or an inpatient admission may be necessary. Thus, we additionally assume that each outbound request r has a specific probability R_r to cause an inbound request during the planning period (e.g., depending on the specific treatment). For outbound requests we still assume that no a priori information is known. This is due to the fact that we do not know in advance from which location such an outbound request would originate or where it's delivery location is. For inbound requests, the patient's home location and the corresponding hospital are already known from the outbound requests. Note, that we also evaluated the case in which we assume to have stochastic information about future outbound requests (regarding their timing and location). Details are described in Section 3.4.5.

3.3 Solution methods

The research described in this chapter aims at studying the effect of using available stochastic information about possible future return transports while planning vehicle routes for the Austrian ambulance service providers. The question is, if stochastic solution algorithms (i.e., taking stochastic information into account while planning) can obtain solutions of higher quality than myopic methods that ignore any stochastic information. For this purpose we propose two conceptually similar pairs of metaheuristic solution methods for the DSDARP with expected return transports. Each pair consists of a myopic approach and its stochastic counterpart. All methods are based on well known metaheuristic concepts from the literature and are adapted to the requirements of the problem at hand. The decision to use pairs of methods is taken in order to allow for direct observation of the effects caused by the additional use of stochastic information. Otherwise, all observed effects could probably be caused by different influence factors (e.g., conceptual differences in the algorithm design). We decide to use two slightly different pairs of algorithms to see if the found effects are also depending on the conceptual basis of the underlying approach or if they are generally applicable.

The first method we adapt to the requirements of the DSDARP with expected return transports is based on a variable neighborhood search (VNS) approach presented by Parragh et al. [9]. This method represents a very efficient and effective state-of-the-art solution approach proposed for the static DARP. The used neighborhood operators are well tested and documented and can therefore be used as a basis for our modifications. We adapt the method in order to handle dynamically arising transportation requests, resulting in a dynamic VNS approach. This first myopic algorithm ignores any available stochastic information about future transportation requests and simply treats them as dynamic.

Based on this dynamic variable neighborhood search method we develop an adaptation of the stochastic VNS (S-VNS) proposed by Gutjahr et al. [38] for stochastic project portfolio analysis. This solution approach takes stochastic information about future inbound transportation requests into account while designing the vehicle routes for the Austrian ambulance service providers. To be precise, the stochastic information is used to evaluate solutions based on sampled future request scenarios whenever two solutions need to be compared. The resulting dynamic S-VNS method is thus the stochastic counterpart to the (myopic) dynamic VNS.

The second pair of metaheuristics consists of adaptations of two methods proposed by Bent and Van Hentenryck [40] for the stochastic vehicle routing problem. We adapt the myopic multiple plan approach (MPA) such that it fits the requirements of the DSDARP. We hereby use our dynamic VNS as a local search component within the MPA framework. Based on this adaptation of the MPA, we build a modified variant of the multiple scenario approach (MSA) for the problem at hand. Similar to S-VNS, MSA also incorporates available stochastic information in the form of sampled future requests in the process of designing the vehicle routes. The information is, however, not just used for comparison purposes, but is directly integrated into the existing solution during the search process. This way we again obtain a myopic base method (MPA) and its corresponding stochastic counterpart (MSA).

3.3.1 Simulation framework

In order to test different scenarios for the dynamic stochastic dial-a-ride problem, we use a simulation framework which is designed as a background service for the solver modules. It supplies the solution algorithms with all the required information on a query basis. This means, that it loads and manages all problem specific information like for example test instance data, distance matrices, vehicle information, and request arrival lists. In addition, the status information for all requests is handled based on the information supplied by the solution algorithms.

At the beginning, the framework loads all problem specific information from file and starts the selected solver module. From this time on, the simulator provides a request interface to the solution algorithm and limits it's activity to reacting on incoming requests through this interface in order to minimize resource consumption. Whenever the solver needs problem specific information (e.g., travel times between two locations), it puts a request to the simulator and receives the required information as an answer. In order to keep the simulation up to date, all relevant status updates are forwarded to the simulator via the interface as well (e.g., departures or arrivals of vehicles). Also, whenever the solution method reaches a point at which it is able to handle new requests, it requests the corresponding list of requests which arose since the last request. At this occasion, the simulation framework updates the simulation time proportional to the actual CPU time that elapsed since starting the solver module. Proportional hereby means that simulation time does not necessarily equal run time. Nevertheless, all results presented in this book are obtained by test runs which are performed in real time.

3.3.2 Initial solution

As all our solution methods are to some extent based on the principles of the well known VNS metaheuristic, they need an initial solution from which the search process can start. For the purpose of finding such a feasible initial solution, we use a modified version of the well known cheapest insertion heuristic proposed by Rosenkrantz et al. [42] for the traveling salesman problem. This ensures, that all solution methods start from the same initial solution and thus none of the solutions has an advantage over the other methods regarding this aspect. We give an outline for this modified method in Algorithm 1. In the pseudo code, we denote by $\Delta(x, r)$ the increase in tardiness caused by inserting a request r into an existing route in solution x and by $\Delta(x^+, r)$ the increase caused by inserting r into a new route in solution x. Furthermore, $V(x)$ represents the number of vehicles which is still unused in a given (partial) solution x.

Algorithm 1 Structure of the modified cheapest insertion heuristic

```
1:  $R \leftarrow$ ListOfKnownRequests()
2:  $x \leftarrow$ AddEmptyRoute()
3:  for $r \in R$ do
4:      if $\Delta(x, r) > \Delta(x^+, r)$ AND $V(x) > 0$ then
5:          $x \leftarrow$ InsertRequestIntoNewRoute($x, r$)
6:      else
7:          $x \leftarrow$ InsertRequestAtBestPosition($x, r$)
8:      end if
9:  end for
10: return $x$
```

Starting from a (partial) solution containing only a single empty route, the method iteratively inserts all known transportation requests into the solution one after another. For each request the algorithm checks all feasible combinations of insertion positions for the pickup service and the delivery service within the current solution. The actual insertion is then performed at the positions which cause the smallest deterioration in solution quality. As an alternative to inserting a request in an existing route, the method also checks the option of inserting the request in a new empty route. This is of course only possible if not all vehicles are already used in the current (partial) solution.

For any pair of insertion positions, the algorithm has to check the feasibility of the resulting solution. This task is performed using a modified version of the well known scheduling algorithm proposed by Hunsaker and Savelsbergh [43] with a reported algorithmic complexity of $\mathcal{O}(n)$. Our modifications enable this algorithm to handle soft time windows (including the depot time window at the end of a day). Also, the scheduling method may not change the timing of already executed services. This means, that all services to which a vehicle has already departed are considered as frozen and must not be modified. If the sequence of services in a route is feasible with respect to vehicle capacity and maximum ride times of the patients, the method returns an efficient scheduling for this route.

3.3.3 Dynamic variable neighborhood search

The concept of the variable neighborhood search (VNS) metaheuristic was proposed by Mladenović and Hansen [39] for static optimization problems. It has since then proven to be a powerful solution approach for numerous problems and was used in a large number of publications during the last years (see [44] for a recent survey).

The basic outline of the traditional VNS method is given in Algorithm 2. We modified this algorithm in order to obtain a solution method capable of solving the DDARP (and thus also the DSDARP), which will be denoted as dynamic VNS in what follows. The outline of dynamic VNS is presented in Algorithm 3. The main difference between the two methods is caused by the need to periodically update the

Algorithm 2 Structure of traditional VNS

```
1: $x \leftarrow$ InitialSolution()
2: $\mathcal{N}(\kappa) \leftarrow$ SelectFirstNeighborhood()
3: **while** StoppingCriterionNotMet() **do**
4:     $x' \leftarrow$ ShakeSolution($x$,$\mathcal{N}(\kappa)$)
5:     $x'' \leftarrow$ LocalSearch($x'$)
6:     $x \leftarrow$ MoveOrNot($x, x''$)
7:     $\mathcal{N}(\kappa) \leftarrow$ SelectNextNeighborhood($\kappa$)
8: **end while**
9: **return** $x$ as best found solution
```

Algorithm 3 Structure of dynamic VNS for the DSDARP

```
1: $x \leftarrow$ InitialSolution()
2: $\mathcal{N}(\kappa) \leftarrow$ SelectFirstNeighborhood()
3: **while** StoppingCriterionNotMet() **do**
4:     $x \leftarrow$ InsertNewRequests($x$)
5:     $x' \leftarrow$ ShakeSolution($x$,$\mathcal{N}(\kappa)$)
6:     $x \leftarrow$ MoveOrNot($x, x'$)
7:     $\mathcal{N}(\kappa) \leftarrow$ SelectNextNeighborhood($\kappa$)
8: **end while**
9: **return** $x$ as best found solution
```

current (partial) solution by inserting new transportation requests which occurred since the last update (Line 4).

Our implementation of dynamic VNS is based on the version of VNS presented by Parragh et al. [9] for the static DARP. We adapted the proposed neighborhood operators in order to handle the dynamic requests. As these neighborhood operators already include some kind of local search behavior, the use of an additional local search phase within the VNS framework did not prove to be beneficial. Therefore, our dynamic VNS algorithm corresponds to the concept of reduced VNS (without local search). Thus, Line 5 of the static version of VNS was omitted in our dynamic version.

The objective of our research is to create efficient vehicle routes for the Austrian ambulance service providers. This means, that each vehicle route represents one working day of a vehicle crew. Therefore, the stopping criterion of our dynamic VNS algorithm is its run time. If the end of the simulated working day (10 hours in our case) is reached, the solution method stops.

Insert new requests

As described before, the conceptual difference between classical VNS and dynamic VNS is caused only by the periodic updating of the current solution by adding new dynamic requests. This can be seen on Line 4 of Algorithm 3. In detail, the method requests a list of new requests from the simulation framework at the beginning of each iteration. If there are requests which occurred since the beginning of the last iteration, they must be inserted into the current (partial) solution, as requests may not be rejected. This insertion is performed using the modified cheapest insertion heuristic which is used to determine the initial solution for all our algorithms (see Section 3.3.2 for details). Hereby, the current (partial) solution is used as an input to the insertion method and all parts of the solution which have already been serviced are considered as frozen. In other words, only insertion positions after the last stop to which a vehicle has already departed can be feasible.

Shake solution

We use a set of highly efficient and effective neighborhood operators proposed for the static DARP by Parragh et al. [9] as the basis of our dynamic VNS. These operators are well tested and documented, which make them ideal candidates for re-using them. Due to the dynamic nature of the DDARP, however, they need to be slightly adapted in a similar way as the previously described cheapest insertion method. Routes which were already (partially) serviced, can only be modified to a limited extent by any neighborhood operation. To be precise, if a vehicle already departed towards a request's pickup location, the complete request (pickup and delivery location) must not be assigned to be serviced by a different vehicle any more. Also, each stop along the route to which a vehicle already departed must not be re-scheduled and no request(s) may be inserted before them.

We use four different neighborhood operators. Each of these operators is used in five different intensity levels $\kappa = \{1 \dots 5\}$, defining the degree of perturbation caused by the operator. The first neighborhood is based on a *move* operator. It randomly selects a single route out of the current (partial) solution and removes κ randomly selected transportation requests from it. These requests are then iteratively re-inserted into the current solution using the modified cheapest insertion heuristic. Each request is inserted into the route and at the position where it causes the lowest deterioration in solution quality. The second neighborhood is defined by a *swap* operator. This operator randomly selects two different routes from the current solution and removes a randomly selected sequence of up to κ consecutive requests from each of them. These requests are then iteratively re-inserted into the other route. This means, a request may only be re-inserted into one of the two selected routes, but not the one from which it was just removed. The third operator in our set is a *chain* operator. It randomly selects two routes: an origin and a destination. It removes a randomly selected sequence of up to κ requests from the origin route and re-inserts these requests into the destination route using the modified cheapest insertion strategy. Then, it uses the destination route as new origin route, randomly selects a new destination route and repeats the described procedure κ times. The last neighborhood structure is based on a *zero split* operator. It randomly selects one route from the solution and determines all positions in the sequence of services at which the vehicle is empty ("zero split points"). Then, it randomly selects two of these points (with up to $\kappa - 1$ other zero split points in between) and removes the sequence of services which is delimited by these two points. Finally, the removed requests are re-inserted into the solution using the cheapest insertion method (without limitation regarding the route into which a request can be re-inserted). Note, that each neighborhood operator is used to randomly create one single solution.

Move or not and select next neighborhood

After perturbing the current incumbent solution by applying one of the described neighborhood operators to it, the dynamic VNS method needs to determine, if the so found solution will be used as a new starting point for upcoming iterations of the search process or not. For this purpose, it compares the current solution and the new candidate solution based on their objective values and accepts the new solution only if it is better than the current incumbent solution. If the found candidate solution is infeasible (with respect to the vehicle capacities or the maximum detour duration), it can never be accepted as a new current incumbent solution.

Besides deciding if the found solution is used as a new current incumbent solution, the algorithm also needs to determine which neighborhood operator to use in the next iteration. This depends on the result of the first decision: if the new solution was accepted, the next iteration will start with the first operator (*move*) and the lowest intensity level, $\kappa = 1$. Otherwise, the intensity level will be increased, until it exceeds the maximum intensity level of $\kappa = 5$. In this case, the next neighborhood operator is used with the lowest intensity level (or if this was the last operator, it starts from the first one again). The sequence is as follows: *move* $\rightarrow$ *swap* $\rightarrow$ *chain* $\rightarrow$ *zero split.*

Time complexity of neighborhood operators

Let $\bar{x} = \{x_1, x_2, x_3, \ldots, x_j\}$ be a solution consisting of j vehicle routes. Furthermore, let $\eta = |x|$ be the number of stops (pickup and delivery) along any of these routes. For a specific neighborhood size κ, we can then determine the worst case time complexity of each neighborhood operator. Note, that the approximate complexity of all used neighborhood operators reduces to the complexity of re-inserting the removed requests into the solution as this is the most complex part.

For the *move* operator, re-insertion in the worst case requires to check each of the $\frac{\eta^2}{2}$ insertion positions for both, the pickup and delivery part of each removed request in each of the j routes in the solution. Additionally, we need to re-schedule each route after the insertion to check for feasibility and to determine the solution quality (which has a complexity of $\mathcal{O}(\eta)$). Therefore, the computational effort is $\kappa j \frac{\eta^3}{2}$ and time complexity is $\mathcal{O}(\eta^3)$. In the case of the *swap* operator, we do not need to check the insertion positions in each route, but only in the two affected routes. Therefore, the computational effort is $2\kappa\frac{\eta^3}{2}$ and time complexity is $\mathcal{O}(\eta^3)$. The time complexity of the *chain* operator is very similar to the one of the *swap* operator. One difference is, that the sequence of requests is only moved to the destination route and not vice versa. Another difference is, that up to κ sequences are moved from one route to another. Therefore, the computational effort is $\kappa^2 \frac{\eta^3}{2}$ and time complexity is $\mathcal{O}(\eta^3)$, again. For the *zero split* neighborhood operator, the actual complexity depends on the sequence length determined by two zero split points. In the worst case, this sequence includes all requests in a route, so κ does not make any difference. As re-inserting the removed requests requires testing all $\frac{\eta^2}{2}$ insertion positions in every route, the computational effort is $j\frac{\eta^4}{2}$ and time complexity is $\mathcal{O}(\eta^4)$. However, an efficient implementation can avoid checking a large number of insertion positions without re-scheduling the corresponding route (e.g., by storing the actual load of each vehicle at each stop) and therefore the actual average case time complexity of our neighborhood operators is by far lower than the worst case time complexity.

3.3.4 Dynamic stochastic variable neighborhood search

Gutjahr et al. [38] proposed a stochastic extension of the variable neighborhood search metaheuristic which is called S-VNS. They demonstrated it's application to project portfolio analysis. An outline of this original S-VNS method is given in Algorithm 4. Its overall structure is similar to the classical VNS metaheuristic. The algorithm starts with constructing an initial solution and then iteratively searches for better solutions until a specific stopping criterion is met.

The most important difference between VNS and S-VNS is the way how solutions are being compared. The classical VNS directly compares a found candidate solution with the current incumbent solution, ignoring any possible stochastic information about the future. In order to exploit this stochastic information about the future, S-VNS utilizes a sampling based comparison procedure instead. This means, that S-VNS uses the stochastic information to sample a set of s_0 possible scenarios of the future. These scenarios are then included in the compared solutions one after another and the resulting solutions are evaluated with respect to their sample average estimator (SAE). The SAE for the expected objective function value is simply the average over all s_0 scenarios. By using the SAE as a base for comparison, S-VNS always selects the solution providing the better average performance with respect to the used set of sampled scenarios. This concept is also used for the (stochastic) local search phase of S-VNS (Lines 7 to 17 in Algorithm 4).

Another conceptual difference to traditional VNS is the fact that S-VNS does not only use a current incumbent solution (denoted as x in Algorithm 4) but also keeps track of the best solution found during the complete search process ($\hat{x}$). In order to compare this solution to the current incumbent solution based on their sample average estimators, a so called *Tournament* step is performed once in each iteration of the algorithm (Lines 22 to 25). In the original S-VNS concept the size s_m of the set of scenarios use for this *Tournament* step is increased continuously in order to prefer more robust solutions towards the end of the search process. Note, that this

Algorithm 4 Structure of traditional S-VNS

```
 1: $x \leftarrow$ InitialSolution()
 2: $\hat{x} \leftarrow x$
 3: $m \leftarrow 1$
 4: $\mathcal{N}(\kappa) \leftarrow$ SelectFirstNeighborhood()
 5: while StoppingCriterionNotMet() do
 6:     $x' \leftarrow$ ShakeSolution($x, \mathcal{N}(\kappa)$)
 7:     $\rho \leftarrow 1$
 8:     repeat
 9:         $Z \leftarrow$ SampleFutureRequests($s_0$)
10:         $x'' \leftarrow$ BestNeighbor($x', Z$)
11:         if SAE($x'', Z$) $\gg$ SAE($x', Z$) then
12:             $x' \leftarrow x''$
13:             $\rho \leftarrow \rho + 1$
14:         else
15:             $\rho \leftarrow \rho_{max} + 1$
16:         end if
17:     until $\rho > \rho_{max}$
18:     $Z' \leftarrow$ SampleFutureRequests($s_0$)
19:     if SAE($x', Z'$) $\gg$ SAE($x, Z'$) then
20:         $x \leftarrow x'$
21:     end if
22:     $Z'' \leftarrow$ SampleFutureRequests($s_m$)
23:     if SAE($x, Z''$) $\gg$ SAE($\hat{x}, Z''$) then
24:         $\hat{x} \leftarrow x$
25:     end if
26:     $\mathcal{N}(\kappa) \leftarrow$ SelectNextNeighborhood($\kappa$)
27:     $m \leftarrow$ SelectNextTournamentSampleSize($m$)
28: end while
29: return $\hat{x}$ as best found solution
```

set of scenarios is not the same as the one used for comparing the current incumbent solution to candidate solutions.

A first intuitive adaptation of the traditional S-VNS concept to the requirements of the dynamic DARP could look like the outline presented in Algorithm 5. The notation of this listing is a more compact one in order to make the algorithm easier to read (e.g., the local search phase of Lines 7 to 17 in Algorithm 4 is written as Line 9 in Algorithm 5). Similar to the adaptations of dynamic VNS, the concept of S-VNS has to be modified in order to handle dynamically occurring transportation requests. For this purpose, the algorithm has to insert all new requests into the current incumbent solution (Line 6) as well as into the best so far solution (Line 7). This straight forward adaptation, however, has a major disadvantage. In the context

Algorithm 5 Structure of modified S-VNS

```
1: $x \leftarrow$ InitialSolution()
2: $\hat{x} \leftarrow x$
3: $m \leftarrow 1$
4: $\mathcal{N}(\kappa) \leftarrow$ SelectFirstNeighborhood()
5: **while** StoppingCriterionNotMet() **do**
6:     $x \leftarrow$ InsertNewRequests($x$)
7:     $\hat{x} \leftarrow$ InsertNewRequests($\hat{x}$)
8:     $x' \leftarrow$ ShakeSolution($x$,$\mathcal{N}(\kappa)$)
9:     $x'' \leftarrow$ LocalSearchSAE($x', s_0$)
10:    $x \leftarrow$ MoveOrNotSAE($x, x'', s_0$)
11:    $\hat{x} \leftarrow$ Tournament($\hat{x}, x, s_m$)
12:    $\mathcal{N}(\kappa) \leftarrow$ SelectNextNeighborhood($\kappa$)
13:    $m \leftarrow$ SelectNextTournamentSampleSize($m$)
14: **end while**
15: **return** $\hat{x}$ as best found solution
```

Algorithm 6 Structure of dynamic S-VNS for the DSDARP

```
1: $x \leftarrow$ InitialSolution()
2: $\mathcal{N}(\kappa) \leftarrow$ SelectFirstNeighborhood()
3: **while** StoppingCriterionNotMet() **do**
4:     $x \leftarrow$ InsertNewRequests($x$)
5:     $x' \leftarrow$ ShakeSolution($x$,$\mathcal{N}(\kappa)$)
6:     $Z \leftarrow$ SampleFutureRequests($1, S_{max}$)
7:     $x \leftarrow$ MoveOrNotSAE($x, x', Z$)
8:     $\mathcal{N}(\kappa) \leftarrow$ SelectNextNeighborhood($\kappa$)
9: **end while**
10: **return** $x$ as best found solution
```

of the DSDARP, the evaluation of a solution based on a set of sampled future transportation requests is a time consuming task. Such an evaluation requires the algorithm to insert all sampled requests of a set into the solution followed by a scheduling algorithm in order to evaluate the solution's quality. Unfortunately, this modified S-VNS algorithm would require large numbers of such evaluations to be performed and thus the performance of the algorithm would not allow to efficiently solve the DSDARP.

As a result, we present another version of the S-VNS metaheuristic, which includes further modifications in order to overcome this problem. The resulting solution method, which we call dynamic S-VNS, is outlined in Algorithm 6. This final modified version of dynamic S-VNS is conceptually more similar to the traditional VNS concept than the first intuitive adaptation. This is an additional advantage, as it keeps the risk of obtaining different results as an effect of conceptual differences between dynamic VNS and dynamic S-VNS as small as possible. The main difference compared to the original S-VNS concept is, that no best so far solution is recorded and thus all related comparisons in the *Tournament* step are omitted. Additionally, dynamic S-VNS uses only a single set of sampled future return transports for the comparison of two solutions in the *MoveOrNotSAE* step (i.e., $s_0 = 1$). The shaking phase of our dynamic S-VNS uses the same set of neighborhood operators in the same order as our dynamic VNS method (see Section 3.3.3 for details). As already mentioned in the section about dynamic VNS, the local search step does not lead to significantly better solution quality and is therefore omitted as well.

Another interesting question that needs to be answered is, if it makes sense to sample all future return transports that might occur until the end of the planning period. The alternative is to introduce a limitation on the time until which the sampled requests are taken into account while planning. Extensive testing showed, that using a look ahead time window S_{max} (see Line 6) for the sampling of return transports is in most cases beneficial to solution quality. This means, that sampled requests are used for planning if the beginning of their pickup time window is not more than S_{max} minutes in the future. Using such a limit has two positive effects on the solution approach. First, it significantly reduces the number of sampled return transports. This leads to a better algorithm performance as the solution evaluations based on these samples are not as computationally expensive as if run without such a limitation. Second, the amount of noise increases with higher values for S_{max} and is highest when no limit is used. This means, the farther a sampled request is in the future, the higher is the risk that a planned solution will be thrown into disarray by additional incoming dynamic requests. However, using a limitation on the look ahead time window also bears the risk of ignoring information that might be useful for planning. In other words, if the value for S_{max} is set too low, this may worsen the obtained solution quality as the algorithm doesn't use enough sampled information any more. Therefore, the value for S_{max} should be carefully tuned as presented in Section 3.4.3.

An intuitive assumption might be that using a limitation on the sampling horizon of $S_{max} = 0$ would reduce the dynamic S-VNS to a myopic dynamic VNS as no more sampled return transports would be used for planning. This, however, is not completely correct. Even if the look ahead time window is set to zero, the set of sampled results might still include some return transports. This is caused by the conceptual design of the sampling procedure. Return transports are sampled for all outbound requests if their corresponding return transports did not arise up to now. This is done even if the underlying distribution would indicate that the return transport should already have occurred (in this case, the beginning of the sample's pickup time window is set to the current time). The reason for this is, that although the distribution indicates that the return transport should already be known by now, it might still occur anyways. Preliminary testing showed that using even this little amount of additional information may in some cases still be enough to obtain slightly better results than with a purely myopic approach.

Algorithm 7 Structure of the MPA for the DSDARP

```
1: $P \leftarrow$ InitialSolution()
2: $\mathcal{N}(\kappa) \leftarrow$ SelectFirstNeighborhood()
3: while StoppingCriterionNotMet() do
4:     $x \leftarrow$ SelectCurrentIncumbent($P$)
5:     for $\bar{x} \in P$ do
6:         if $(\bar{x} \neq x) \wedge ($Timeout$(\bar{x}, x) \vee$ Departure$(\bar{x}, x))$ then
7:             $P \leftarrow P \setminus \{\bar{x}\}$
8:         else
9:             InsertNewRequests($\bar{x}$)
10:        end if
11:    end for
12:    $x' \leftarrow$ ShakeSolution($x, \mathcal{N}(\kappa)$)
13:    if $x' \notin P$ then
14:        $P \leftarrow P \cup \{x'\}$
15:    end if
16:    $\mathcal{N}(\kappa) \leftarrow$ SelectNextNeighborhood($\kappa$)
17: end while
```

3.3.5 Multiple plan approach

The concept of the multiple plan approach (MPA) was proposed by Bent and Van Hentenryck [40] for the dynamic vehicle routing problem with time windows (VRPTW). The design of the MPA is a generalization of the work by Gendreau et al. [45] who proposed a parallel tabu search algorithm arranged around an adaptive memory containing multiple solutions. It is in fact a metaheuristic solution framework designed to be combined with almost any search component. In the original publication, the authors used a large neighborhood search approach to demonstrate the method.

The main idea of the MPA is to maintain a set of solutions found during the search process in order to preserve as much flexibility for later planning decisions as possible. This means that starting from a given set of alternative solutions for the static part of a dynamic vehicle routing problem, the MPA continuously searches for further alternative solutions while time progresses. Whenever a new alternative solution is found, it is inserted into the set of solutions. As the MPA is designed to solve dynamic problems, all solutions in the set need to be kept compatible with the decisions made until the current time. For this purpose, the framework maintains a single current incumbent solution which is used to determine the actual decisions made. This master solution is selected out of all solutions in the set and is re-selected whenever a new solution is found and stored in the set. Based on the decisions made in the current incumbent solution, the MPA eliminates solutions from the set if they are no longer compatible with the master solution (i.e., if a vehicle departs in the current incumbent solution, but not in another solution or vice versa). Additionally, new transportation request are inserted into all solutions in the set and solutions which cannot accommodate a request are eliminated from the set (unless no solution can accommodate the requests which would in this case be rejected).

The multiple plan approach is a concept which is already designed to handle vehicle routing problems with dynamic requests and thus we expect it to be suitable for the dynamic DARP as well. As Bent and Van Hentenryck state that the MPA framework is independent of the used search component, we decide to use our implementation of dynamic VNS for this purpose. By doing so, we also guarantee that found differences between our solution methods are not caused by a different local search technique, but are due to the relevant design decisions we make.

The outline of the MPA framework is presented in Algorithm 7. Like our other local search based metaheuristic approaches, it starts by constructing an initial solution (Line 1). For this purpose, our adaptation of the cheapest insertion heuristic is used (see Section 3.3.2 for details). After this, the algorithm iteratively searches for new alternative solutions. At the beginning of each iteration it selects the current incumbent solution out of the current set of solutions (Line 4). Then, all incompatible solutions are removed from the set of solutions (Line 7) and new transportation requests are inserted into all remaining solutions in the set (Line 9). As no requests may be rejected in the case of our DSDARP, the MPA does not need to eliminate any solutions from the set at this point (as would be the case in the original MPA concept). The search component used to find new alternative solutions (Line 12) is the same as the one used for our implementation of dynamic VNS. This means, that the same neighborhood operators are used in the same way as for dynamic VNS. For this reason, again an additional local search phase does not bring an additional benefit and is thus omitted. If the found solution is not already known, it is stored in the set of solutions, otherwise it is discarded (Line 13). Finally, the method decides which neighborhood to use next (Line 16; following the same rules as in the case of dynamic VNS) and continues with the next iteration.

3.3.6 Multiple scenario approach

The idea of the multiple scenario approach (MSA) was presented as a logical extension to the multiple plan approach by Bent and Van Hentenryck [40]. It is a stochastic version of the MPA and was demonstrated for the dynamic VRPTW. While the MPA ignores any available stochastic information about the studied problem, the idea for the MSA is to exploit this information while planning the vehicle routes. This is, however, done in a slightly different way than in the case of (dynamic) S-VNS. Sampled future requests are not only used to compare two different solutions but also during the search phase itself.

The MSA is (like the MPA) a metaheuristic framework that is designed to work independent from the underlying search procedure. Therefore, we again use our implementation of dynamic VNS for this purpose. By this we again guarantee that all found effects are due to the exploitation of the stochastic information and not

Algorithm 8 Structure of the MSA for the DSDARP

```
1: $P \leftarrow$ InitialSolution()
2: $\mathcal{N}(\kappa) \leftarrow$ SelectFirstNeighborhood()
3: **while** StoppingCriterionNotMet() **do**
4:    $x \leftarrow$ SelectCurrentIncumbent($P$)
5:    **for** $\bar{x} \in P$ **do**
6:       **if** $(\bar{x} \neq x) \wedge$ (Timeout$(\bar{x}, x) \vee$ Departure$(\bar{x}, x)$) **then**
7:          $P \leftarrow P \setminus \{\bar{x}\}$
8:       **else**
9:          InsertNewRequests($\bar{x}$)
10:      **end if**
11:   **end for**
12:   $Z \leftarrow$ SampleFutureRequests($1, S_{max}$)
13:   $x' \leftarrow$ AddSampledRequestsToSolution($x, Z$)
14:   $x' \leftarrow$ ShakeSolution($x', \mathcal{N}(\kappa)$)
15:   $x' \leftarrow$ RemoveSampledRequestsFromSolution($x', Z$)
16:   **if** $x' \notin P$ **then**
17:      $P \leftarrow P \cup \{x'\}$
18:   **end if**
19:   $\mathcal{N}(\kappa) \leftarrow$ SelectNextNeighborhood($\kappa$)
20: **end while**
```

cause by any other conceptual differences between the MPA and the MSA.

An outline of the multiple scenario approach is presented in Algorithm 8. The structure is very similar to the one of the MPA. The algorithm starts by creating an initial solution using the modified cheapest insertion method presented in Section 3.3.2 followed by an iterative search phase. At the beginning of each iteration, the current incumbent solution is selected out of the set of solutions (Line 4). After that, all solutions which are no longer compatible with the current incumbent solution are removed from the set (Line 7). New requests are inserted into all remaining solutions in the set using the modified cheapest insertion method (Line 9). The main difference to the MPA is, that future requests are sampled and inserted into the current incumbent solution before the shaking step (Lines 13 and 14). After the shaking these sampled requests are removed from the solution again (Line 15). This is done in order to obtain a solution including gaps in the schedule of the request, which can at a later point in time be used to accommodate real future requests more easily. The resulting solution is then stored in the set of solutions if it was not known before (Line 16) and the next neighborhood operator is selected accordingly (Line 19). Hereby, the same neighborhood structures are used as with all our other methods.

The multiple scenario approach as well as the multiple plan approach use a set of solutions found during the search process. This leads to the requirement of using a single solution to determine which decisions should finally be made while the day progresses. Therefore, at the beginning of each iteration both methods select a current incumbent solution out of all solutions in the pool. According to Bent and Van Hentenryck [40], the best way to perform this selection is by using a consensus function which is a strategy similar to the concept of least commitment. Interestingly, in the case of the dynamic stochastic dial-a-ride problem, this is not necessarily true. In fact, our extensive tests show that altering the original design of the MPA and the MSA in order to always select the currently best solution out of the set of solutions leads to better results than using the proposed consensus function. Also, using the so selected current incumbent solution as a basis for the search process seems to be the favorable alternative for this problem.

3.4 Computational experiments

In order to study the effect of exploiting the available stochastic information about future return transports while planning the vehicle routes for the Austrian ambulance service providers, we create a set of real world based test instances. Starting from a set of real world data provided by one of the Austrian ambulance service providers about daily operations in an Austrian city during one year as well as a real world street network, we derive our test instances in order to represent one 10 hour working day each. This is done by extracting the distribution parameters for the interarrival times of the requests serviced during this period. The details of this analysis are described in the master thesis of Kritzinger [46] and a short summary of the process is presented in Section 3.4.1. This information about the underlying stochastic distributions is used to sample transportation requests for each test instance as described in Section 3.4.2. All computational experiments are performed on one core of a SUN Fire X2270 server with 2 quad-core Intel Xeon X5550 processors (2.66 GHz) with 24 GB of shared memory. The algorithms are implemented using C++ and the GNU compiler g++ in its version 4.1.2 on CentOS 5.5.

3.4.1 Data generation

This section is a short summary of the data analysis performed by Kritzinger [46] in her master thesis. The underlying real world data set was supplied by an Austrian ambulance service provider and consists of log entries for a total of 125,035 anonymized transportation requests performed during the year 2004. The remainder of this section is taken from [17].

The first observation of the analysis was that approximately 50% of all transportation requests are already known in the morning (static). The other 50% arises during the day dynamically. To obtain information about these dynamic requests, the interarrival times of the sample were calculated and filtered. Then, the day was split into segments of one hour. For each segment, the number of interarrival times that fall into a specific interval (e.g., 0 to 10 minutes, 10 to 20 minutes, ...) was counted. The analysis indicated an exponential distribution. This assumption was tested using χ^2 tests. The null hypothesis was, that the data is exponentially distributed with a continuous density $f(x) = \lambda e^{-\lambda x}$ for $\lambda > 0$ and an expected value of $\mathrm{E}(X) = \frac{1}{\lambda}$. The parameter λ was determined using the maximum likelihood estimation of the reciprocal of the sample's average value. The χ^2 tests returned positive results for approximately 70% of all cases (i.e., do not reject the null hypothesis). For the time between the occurrence of a transportation request and the corresponding latest arrival time at the hospital, results suggest a gamma distribution with a continuous density $f(x) = \frac{\lambda}{\Gamma(\alpha)}(\lambda x)^{\alpha-1}e^{-\lambda x}$, whereby $\alpha, \lambda > 0$, the expected value is $\mathrm{E}(X) = \frac{\alpha}{\lambda}$, and the variance is $\mathrm{Var}(X) = \frac{\alpha}{\lambda^2}$. The values for α and λ were estimated using the generalized method of moments, which proved to be a good estimation.

The analysis continued by taking a look at intervals of 15 minutes. For each of these intervals, the number of working days showing 0, 1, 2, 3, ... return transports was counted. This counting process showed the characteristics of a Poisson process within each of the 15 minute intervals. To verify if the underlying data really follows a Poisson distribution, again χ^2 tests were used. The null hypothesis was that the distribution follows a Poisson distribution with a discrete distribution density of $\mathrm{P}(X = k) = \lambda^k \frac{e^{-\lambda}}{k!}$ with $k \in \mathbb{N}_0$ and an expected value of $\mathrm{E}(X) = \lambda$. Maximum likelihood estimation of the sample's average value was used to determine the parameter λ. The χ^2 tests returned positive results for about 80% of the intervals.

The same procedure was used to determine the distribution of the time between a transportation request's latest arrival time at the hospital and the arrival time of the corresponding return transport. The null hypothesis was that the underlying distribution is a gamma distribution with a continuous density function. The values for parameters α and λ were determined using the generalized method of moments. As the χ^2 test rejected the null hypothesis for the determined parameters, they were iteratively modified to fit the underlying data more precisely. Finally, analysis showed that data suggests that approximately 50% of all transports towards a hospital cause a corresponding return transport in the opposite direction on the same day.

Knowing the statistical distribution parameters, all information required for sampling artificial transportation requests was available. By sampling a set of such transportation requests, a real world inspired test instance could be generated. Also, modifying the distribution parameters allowed for simulating different scenarios of reality (e.g., larger cities). Additionally, the data set reported customer locations as often as they occurred during the observation period. This means, that if the same patient was transported to the hospital 10 times during this period, there existed 10 entries. Using this information enabled us to map the real world geographical distribution of patient locations and the corresponding frequencies to our test instances. This was done by assigning customer locations using a uniform random selection to each artificial request.

	R		
N	30%	50%	80%
100%	149.93	168.73	208.13
200%	281.87	325.07	389.27
300%	404.80	474.33	566.53
400%	523.40	614.47	718.73

Table 3.1: Average number of transportation requests per instance class. (R - return transport probability, N - relative size of the instance set)

	outbound			
R	static	dynamic	inbound	total
30%	135.98	127.33	76.68	340.00
50%	137.68	127.08	130.88	395.65
80%	135.62	128.50	206.55	470.67

Table 3.2: Average number of requests per type depending on return transport probability. (R - return transport probability)

3.4.2 Test instances

Based on the distribution parameters derived from the real world data set (see Section 3.4.1), we create 12 sets of test instances each containing 15 instances. For this purpose we start by sampling the found distribution for the outbound transportation requests (from a patient's home location to a hospital). Beginning at time $t = 0$, the first request's arrival time a_1 is calculated by sampling the interarrival time of the outbound requests. All following requests' arrival times are then determined based on the previous request's arrival time a_{i-1} until the end of the planning horizon (10 hours) is reached. This way we obtain the arrival times of all outbound requests. Based on these arrival times, we calculate the time at which each patient needs to be at the hospital (i.e., the end of the delivery time window) by sampling the corresponding distribution. The delivery time window is set to 30 minutes length. Then, the end of the 30 minute pickup time window can be calculated as the starting time of the delivery time window minus the time required to travel directly from the pickup to the delivery location. Finally, half of the generated requests is set to be static by assigning an occurrence time of $a_r = 0$ to them.

In a second step, the return transports (inbound requests) for the generated outbound requests are generated. For this purpose, a return transport for each of the requests (both static and dynamic ones) is generated with a probability R. The occurrence time a_r for these inbound requests was determined by sampling the distribution for the time between the outbound request's end of the delivery time window and the beginning of the inbound request's pickup time window (i.e., the time at which the patient can be picked up at the hospital again). From this time on, the pickup time window has a length of 60 minutes. The beginning of the inbound request's delivery time window is calculated by adding the time required to travel directly from the hospital to the patient's home location. The length of the delivery time window is set to 90 minutes, which is 60 minutes plus the maximum length of an allowed detour (30 minutes).

In order to cover a larger range of scenarios, the parameters for the interarrival time between the occurrence of two outbound requests as well as the probability for a return transport (R) is modified for each set of scenarios. The former is done to obtain instances with different numbers of outbound requests during the planning period. The idea behind this is to test if larger instances (i.e., representing larger cities) does have and influence on the observed effects. The relative number of outbound requests N is therefore altered in the range of 100% to 400% of the real world size in steps of 100%. The probability for return transports R is set to 30%, 50% and 80%, respectively. Although the real world data suggests that approximately 50% of all outbound requests cause an inbound request on the same day, this is done in order to study if this factor of uncertainty has an influence on the found effects. By using every possible combination of N and R, we obtain 12 different sets of instances. We do not take inbound requests that are not caused by an outbound request on the same day into account. This is because we are primarily interested in the effect of the exploitation of the available stochastic information about future inbound requests. As we do not have any information about such independent requests, they would be treated as being purely dynamic and thus no influence on the obtained effects is expected. The average number of requests per instance depending on the probability for return transports R and the relative instance size N is given in Table 3.1. The average number of requests per instance depending on the probability for return transports and the type of request (static, dynamic, return transports) is presented in Table 3.2.

To obtain a reasonable limit for the number of vehicles available for each of the instances, we create a greedy solution for each of them using the modified cheapest insertion heuristic described in Section 3.3.2 (assuming all requests to be known a priori). The number of vehicles used in each of these solutions is then increased by 10% and used as limit on the number of vehicles available for the respective instance.

3.4.3 Parameter settings

As the main focus of this work is to study the effects and influence factors of exploiting stochastic information about future return transports while planning the vehicle routes for Austrian ambulance service providers, we need solution approaches that are able to solve the dynamic dial-a-ride problem in an efficient and effective way. The existing literature provides a large number of state-of-the-art metaheuristics for different types of vehicle routing problems and also the DARP. Therefore, we decide to base our work on some of the proposed approaches that are well tested and documented. We use the conceptual design and parameter settings suggested by Parragh et al. [9] for the static DARP as a local search basis for all our methods. Additionally, we use the ideas presented by Gutjahr et al. [38] and Bent and Van Hentenryck [40] with some modifications. Although we can base our work on this already available information, we still need to tune some parameters for our solution approaches.

In this part of our work, we use a parameter τ as a threshold value for opening new routes when inserting new requests using the modified cheapest insertion heuristic. Opening an additional route for the new request is considered only if the increase in tardiness by inserting the request in an existing route would be larger than τ. After extensive testing we observe that values between 20 and 30 minutes lead to the best results. As all our methods would be equally affected by this parameter, we decided to fix it to 25 minutes.

Another parameter that needs to be tuned carefully is the look ahead time window S_{max}. The value selected for S_{max} defines the extent to which sampled future requests are taken into consideration by our stochastic solution approaches. This means, that sampled requests that will occur more than S_{max} minutes in the future will be ignored by the methods. It turns our that this parameter has a strong influence on the obtained results. Therefore, we try to find preferable values for this factor by fixing it to a specific value and evaluating the resulting solution quality for all 12 sets of test instances. By comparing the average solution quality over all test instances obtained with different values for S_{max} we are able to identify the most promising values. Detailed results are presented in Section 3.4.4.

Additionally, we studied the effect of a speedup factor of 10 for the simulation time. As mentioned in the description of our simulation framework (see Section 3.3.1), simulation time does not necessarily elapse equally fast as real time. In other words, speeding up simulation time by a factor of 10 causes the simulation to run 10 times faster than real time. After extensive testing we observed, that results obtained using a speedup factor of 10 are not representative for the results achieved in real time. The positive effect of additional run time is much stronger for our dynamic S-VNS and dynamic VNS approaches than for the MPA and the MSA. For this reason, the results obtained with the speedup factor would lead to different conclusions and should therefore not be used for parameter tuning or similar activities.

3.4.4 Results

Look ahead period

The look ahead time window S_{max} has, as described in the previous section, a strong influence on the obtained solution quality of our stochastic algorithms. In order to tune this parameter, we solved all 180 test instances (12 sets with 15 instances each) using values of $S_{max} = \{3, 5, 10, 20\}$ minutes. The obtained average solution quality over all instances indicates, that shorter look ahead time windows seem to be

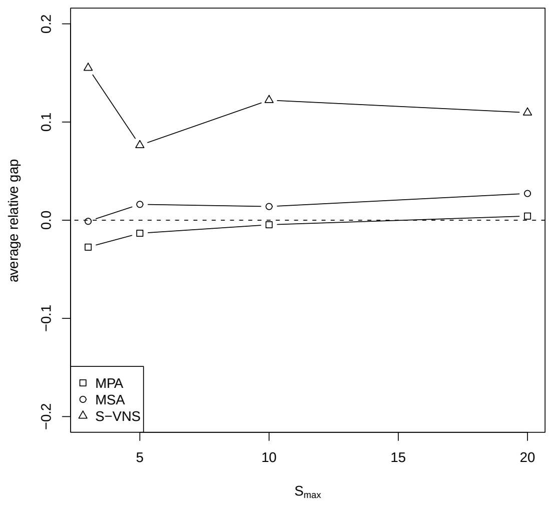

Figure 3.1: Average relative gaps depending on S_{max}.

beneficial for our stochastic approaches. A summary of the average solution quality over all instances is presented in Table 3.5. The values shown in this table are the gaps between the solution obtained by the respective algorithm and the one obtained by dynamic VNS. A negative value indicates superior solution quality obtained by dynamic VNS; a positive value indicates a superior result obtained by the other method. As our objective function is lexicographic, the comparison can only be based on the primary objective (tardiness, gap_1) while the secondary (number of vehicles, gap_2) and tertiary objective (total route duration, gap_3) are reported for the sake of completeness only. Figure 3.1 presents a graphical representation of these results. This picture shows a peak at the value of 3 minutes for dynamic S-VNS and at 20 minutes for the MSA. However, it seems that the MSA is not influenced by the look ahead time window as strongly as dynamic S-VNS. These results lead us to the conclusion that taking only the very near future into account when exploiting the stochastic information about return transports seems to be most beneficial to solution quality. A reason for this might be that newly incoming dynamic requests cause the sample based estimation of a solution's quality to change very frequently. A summary of all solutions obtained using $S_{max} = 3$ and $S_{max} = 20$ is given in Tables 3.3 and 3.4, respectively.

As can be seen from the results presented in Table 3.5, when using a look ahead time window of 3 minutes dynamic S-VNS can obtain solutions which are on average over all test instances 15.50% better than the ones obtained by dynamic VNS. Also, the MSA can improve the average solution quality by 2.72% when using $S_{max} = 20$ minutes. This means, that we are able to verify the hypothesis that taking into

		gap_1 in %			gap_2 in %			gap_3 in %		
R	N	MPA	MSA	S-VNS	MPA	MSA	S-VNS	MPA	MSA	S-VNS
30	100	-1.98	3.81	**13.57**	0.00	-3.03	-3.03	-0.55	-3.53	-36.40
30	200	-0.96	-0.91	**22.75**	-4.83	-6.55	-1.38	-2.00	-1.04	-38.39
30	300	-2.48	-1.19	**40.31**	-6.87	-6.36	-1.78	0.55	-3.36	-35.21
30	400	-5.32	-3.17	**29.03**	-7.05	-5.71	-0.19	0.31	-5.89	-29.62
50	100	10.69	-4.12	**11.89**	-0.60	-3.61	-9.04	-0.74	-4.53	-35.58
50	200	-5.90	-0.50	**36.33**	-7.14	-5.00	-12.50	1.11	-4.83	-40.39
50	300	-7.33	**-3.30**	-5.23	-5.06	-4.10	-3.37	0.65	-6.12	-33.84
50	400	1.57	3.79	**10.27**	-7.24	-7.05	-7.05	-0.53	-5.53	-30.18
80	100	-2.82	5.41	**14.79**	1.14	0.57	-13.14	0.26	-4.12	-36.23
80	200	-4.77	-1.59	**20.76**	-2.30	-3.62	-12.83	-0.17	-5.71	-38.97
80	300	-4.21	**9.57**	5.70	-1.15	-2.75	-11.24	0.40	-7.56	-35.78
80	400	-9.22	-6.99	**4.11**	-5.12	-6.40	-12.80	0.15	-8.94	-33.70
avg.		-2.73	0.07	**17.02**	-3.85	-4.47	-7.36	-0.05	-5.10	-35.36

Table 3.3: Summary of results obtained with $S_{max} = 3$. (R - return transport probability, N - relative instance size)

account stochastic information about future return transports while planning vehicle routes for the Austrian ambulance service providers is beneficial to solution quality. Also, we observe that the design of the used methods does have a strong impact on the obtained advantages of using stochastic information. Knowing that, we want to determine additional factors that influence the amount of improvement that can be achieved. Therefore, we need to examine solution quality in more detail. In what follows, if not explicitly mentioned otherwise, we will refer to solutions obtained using a value of $S_{max} = 20$ for the MSA and $S_{max} = 3$ for S-VNS. Note that the MPA, which is not influenced by S_{max}, achieves an average gap in tardiness of 0.08% over all runs included in Table 3.5. This indicates, that the used long term memory concept (i.e., the use of a set of solutions) does not lead to significant improvements over dynamic VNS in the case of the dynamic stochastic dial-a-ride problem with expected return transports.

Discussion of findings

The detailed results obtained in our computational experiments are shown in Table 3.6 (depending on the relative instance size N) and Table 3.7 (depending on the return transport probability R). The results presented in Table 3.6 show, that dynamic S-VNS obtains a significantly larger improvement over dynamic VNS than the MSA. This advantage is relatively stable but decreases slightly with increasing instance size. The solutions achieved by dynamic S-VNS are between 11.12% and 25.86% better than the ones found by dynamic VNS, depending on the instance size. The multiple scenario approach yields improvements of -1.35% to 9.98% compared

		gap_1 in %			*gap_2* in %			*gap_3* in %		
R	N	MPA	MSA	S-VNS	MPA	MSA	S-VNS	MPA	MSA	S-VNS
30	100	1.60	6.83	**13.23**	-1.20	-1.20	0.00	-1.99	-2.89	-36.30
30	200	-6.16	-7.79	**15.83**	-3.39	-3.39	-1.02	0.47	-4.72	-40.79
30	300	5.29	0.25	**41.95**	-4.57	-5.08	-1.78	-1.40	-5.47	-34.83
30	400	-2.52	8.59	**29.12**	-6.10	-5.90	-0.38	-1.66	-6.41	-31.18
50	100	**7.61**	4.55	4.66	-1.80	-1.80	-8.38	-0.84	-4.01	-38.67
50	200	4.51	10.56	**30.39**	-6.09	-5.02	-12.54	-0.89	-5.37	-39.99
50	300	**3.92**	-0.06	-15.01	-5.83	-6.07	-6.80	0.13	-7.23	-34.17
50	400	-4.93	**1.46**	-0.60	-7.24	-7.24	-5.14	-0.04	-5.21	-28.93
80	100	3.32	15.90	**19.83**	-3.43	-1.14	-15.43	-0.76	-5.94	-36.89
80	200	-5.48	-6.31	**-4.34**	-2.27	-2.92	-15.26	1.66	-5.32	-40.78
80	300	-3.40	2.96	**8.93**	-3.00	-6.22	-16.13	1.40	-8.79	-35.13
80	400	0.10	-2.67	**5.94**	-3.78	-5.22	-17.99	0.79	-9.34	-32.75
avg.		0.32	2.86	**12.49**	-4.06	-4.27	-8.40	-0.26	-5.89	-35.87

Table 3.4: Summary of results obtained with $S_{max} = 20$. (R - return transport probability, N - relative instance size)

	gap_1 in %		*gap_2* in %		*gap_3* in %	
S_{max}	MSA	S-VNS	MSA	S-VNS	MSA	S-VNS
3	-0.11	**15.50**	-4.43	-7.45	-5.24	-35.42
5	1.61	**7.63**	-4.42	-7.42	-5.21	-36.03
10	1.40	**12.22**	-3.99	-7.73	-5.44	-36.23
20	2.72	**10.95**	-4.26	-8.55	-2.72	-10.95

Table 3.5: Average gap in solution quality compared to dynamic VNS obtained with different settings for S_{max}. Positive (negative) values indicate that this method performed better (worse) than dynamic VNS.

to dynamic VNS.

We also test the original design of dynamic S-VNS as proposed in the article of Gutjahr et al. [38] which uses multiple samples for comparison and maintains a best so far solution which is compared to the current incumbent solution based on a different set of samples with an increasing number of samples. This variant performs significantly worse for our test instances as it achieves improvements of only 4.72% to 20.61% compared to dynamic VNS while our modified version obtains between 11.12% and 25.86% improvement.

An interesting observation regarding the effect of the return transport probability R on the solution quality obtained by dynamic S-VNS can be made when looking at Table 3.7. This table shows the average results over all test instances depending on R. The same effect can also be seen in Figure 3.3 which shows the gaps obtained by

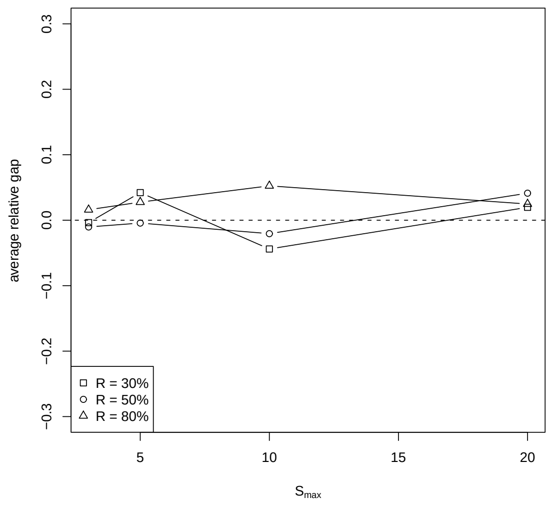

Figure 3.2: Average relative gaps depending on S_{max} and R for the MSA.

dynamic S-VNS compared to dynamic VNS with respect to S_{max} and R. The lower the return transport probability is, the better are the results obtained by dynamic S-VNS compared to dynamic VNS and vice versa. The MSA, however, seems to be far less sensitive to changes in the return transport probability (see Figure 3.2). Intuitively, this effect would be expected to work in the opposite direction. A higher probability for return transports (i.e., less insecurity in this aspect) should imply more accurate samples with respect to future return transports (as it is less likely to sample return transports that in fact will never occur or not to sample return transports that will in fact occur). These more accurate samples should then also lead to more accurate solutions and thus a higher solution quality. In reality, however, the higher return transport probability increases the absolute number of return transports and by that also the total number of dynamic requests relative to the total number of requests in an instance (see Table 3.2). This implies a higher total degree of dynamism which seems to be compensating for the positive effects of the more accurate samples. Thus, the relative advantage of dynamic S-VNS gets lower as the results obtained by the (myopic) dynamic VNS is better at handling highly dynamic situations.

Yet another observation can be seen in Table 3.6. Dynamic S-VNS leads to solutions that use on average 7.45% more vehicles and have 35.42% longer total route durations compared to the solutions found by dynamic VNS. The MSA on average achieves 4.26% and 6.01% higher values than dynamic VNS, respectively. This is not only true for scenarios in which the main objective (i.e., tardiness) is improved compared to dynamic VNS, but for most other scenarios as well. This tradeoff

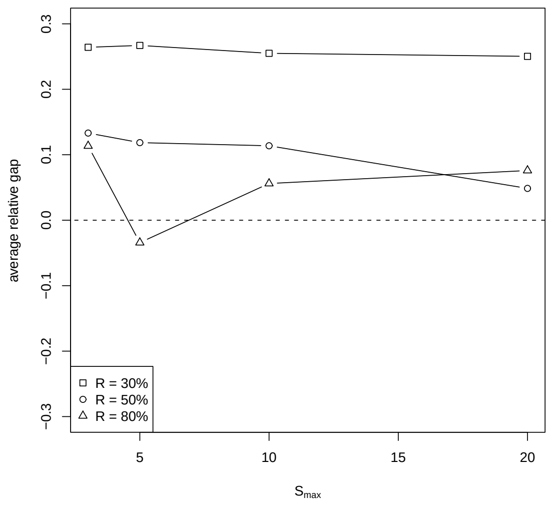

Figure 3.3: Average relative gaps depending on S_{max} and R for dynamic S-VNS.

between tardiness and number of used vehicles (which to some extent induces the higher total duration) was not unexpected. However, the magnitude of this effect is remarkable. We assume it is caused by the relatively small number of iterations performed by dynamic S-VNS and the MSA. It seems that dynamic VNS spends much more time locally optimizing known solutions with respect to the secondary and tertiary objectives (i.e., number of vehicles used and total route duration) than dynamic S-VNS and the MSA.

	gap_1 in %			gap_2 in %			gap_3 in %			iterations			
N	MPA	MSA	S-VNS	MPA	MSA	S-VNS	MPA	MSA	S-VNS	VNS	MPA	MSA	S-VNS
100%	3.53	9.98	**13.31**	0.20	-1.38	-8.50	-0.30	-4.43	-36.07	275,531,013	197,609,100	1,388,240	1,716,533
200%	-4.30	-1.35	**25.86**	-4.69	-3.74	-8.92	-0.30	-5.16	-39.26	77,475,788	56,088,319	490,501	577,590
300%	-4.61	1.43	**11.72**	-4.26	-5.81	-5.63	-0.53	-7.32	-34.98	40,055,460	27,533,506	267,885	318,721
400%	-5.59	0.83	**11.12**	-6.45	-6.10	-6.76	-0.03	-7.15	-31.37	24,391,153	18,007,290	181.636	221,579
avg.	-2.74	2.72	**15.50**	-3.80	-4.26	-7.45	-0.02	-6.01	-35.42	104,363,353	74,809,554	582,065	708,606

Table 3.6: Average solution quality depending on relative test instance size with $S_{max} = 20$ for the MSA and $S_{max} = 3$ for dynamic S-VNS. (N - relative instance size)

	gap_1 in %			gap_2 in %			gap_3 in %			iterations			
R	MPA	MSA	S-VNS	MPA	MSA	S-VNS	MPA	MSA	S-VNS	VNS	MPA	MSA	S-VNS
30%	-2.68	1.97	**26.42**	-4.69	-3.89	-1.60	-0.42	-4.87	-34.90	139,609,711	99,770,945	702,520	735,365
50%	-0.24	4.13	**13.31**	-5.01	-5.03	-7.99	0.12	-5.45	-35.00	102,378,472	74,239,400	551,509	720,376
80%	-5.25	2.47	**11.34**	-1.86	-3.88	-12.50	0.16	-7.35	-36.17	71,101,879	50,418,317	492,169	670,076
avg.	-2.72	2.86	**17.02**	-3.85	-4.27	-7.36	-0.05	-5.89	-35.36	104,363,354	74,809,554	582,066	708,606

Table 3.7: Average solution quality depending on return transport probability with $S_{max} = 20$ for the MSA and $S_{max} = 3$ for dynamic S-VNS. (R - return transport probability)

		gap$_1$ in %		*gap*$_2$ in %		*gap*$_3$ in %	
R	N	MSA	S-VNS	MSA	S-VNS	MSA	S-VNS
30	100	5.08	**17.63**	-4.27	-1.83	-1.95	-33.20
30	200	7.62	**24.31**	-3.40	-0.68	-1.86	-32.91
30	300	-2.11	**27.46**	-6.84	-1.01	-1.61	-31.19
30	400	-1.38	**28.87**	-6.83	-0.76	-1.33	-28.17
50	100	-1.78	**8.18**	-4.29	-9.20	-1.00	-28.89
50	200	9.46	**12.82**	-6.50	-7.58	-1.06	-32.38
50	300	**-9.63**	-17.65	-4.55	-3.35	-4.16	-29.31
50	400	-3.86	**6.16**	-6.62	-4.35	-1.90	-27.24
80	100	13.18	**32.54**	0.00	-10.06	-2.66	-26.37
80	200	-1.58	**9.33**	-3.93	-8.52	-1.60	-25.98
80	300	**-6.37**	-12.49	-3.42	-11.19	-1.74	-24.88
80	400	-3.96	**-1.41**	-5.80	-13.41	-2.28	-23.20
avg.		0.39	**11.31**	-4.70	-5.99	-1.93	-28.64

Table 3.8: Summary of results obtained with $S_{max} = 3$ in the unbiased test case. (R - return transport probability, N - relative instance size)

3.4.5 Testing for bias

The fact that our stochastic solution approaches only use samples of future inbound requests (from hospital back to a patient's home location) and do not sample future outbound requests (from a patient's home location to a hospital) induces a specific bias. As the algorithms try to accommodate the sampled inbound requests, they might end up forcing vehicles to stay close to hospitals where such requests are expected to occur. This, however, can have contra-productive effects on the ability to react on future outbound requests, thus leading to poorer solution qualities. Note, that this bias could also be a possible explanation for the fact that the original concept of using a consensus function to determine the current incumbent solution does not have the same effect in our case as in the case studied by Bent and Van Hentenryck [40]. The consensus over several biased solutions might strengthen the bias even more, thus leading to lower solution quality.

To verify if this effect might be the reason for the observed deterioration of solution quality of dynamic S-VNS in cases with a higher return transport probability R, we test variants of the stochastic algorithms which also sample outbound requests. To do so, we provided the methods with a list of all patient locations included in our real world data set including the occurrence frequencies. Based on this information, the sampling of future outbound requests is performed the same way as the test instances were created (see Section 3.4.2).

The results for these unbiased tests are presented in Table 3.8. They show, that dynamic S-VNS does perform significantly worse in this unbiased case and MSA performs almost equally in both cases. This result may have two reasons. Either the assumed bias does not cause the effect that R has on the solution quality achieved by dynamic S-VNS or the quality of the sampled outbound requests is poor and therefore further misleads the search process. The latter could be the case as these samples do not only incorporate insecurity regarding the time of occurrence (as do the samples of inbound request) but also regarding the geographical location of upcoming requests. Therefore each of the sampled requests has a very low probability to coincide with a real request. Because of this, the estimation of a solution's quality by dynamic S-VNS might be too far off the truth, thus leading to worse results in the end. The reason why the MSA does not seem to be affected by this in the same extent may be that (other than dynamic S-VNS) it does not discard a solution based on the solution quality regarding the used sample but stores it in the pool anyway. Therefore, the MSA has a better chance to correct for the bad sampling at a later point in time, which dynamic S-VNS has not.

3.5 Summary

In this chapter we study the effects of exploiting stochastic information about future return transports while planning the vehicle routes for an Austrian ambulance service provider. For this purpose, we compare four different adaptations of metaheuristic solution methods for the dynamic stochastic dial-a-ride problem with expected return transports. We are able to identify circumstances that are beneficial for the proposed stochastic approaches and some which are not. Especially if the number of return transports is relatively low compared to the total number of transportation requests, dynamic S-VNS strongly outperforms the myopic methods (dynamic VNS and the MPA).

Another interesting finding is the influence which the look ahead time window's size has on the solution quality obtained by our stochastic solution methods. It turns out that taking only a very short period of the future into account while planning brings the largest benefit. Our results indicate that the length of this period should not exceed 20 minutes in order to obtain the best possible results. The achieved average gaps to solutions obtained by dynamic VNS reach up to 15.90% for the MSA and up to 41.95% for dynamic S-VNS. However, in the worst case, average gaps can drop to -7.79% for the MSA and -15.01% for dynamic S-VNS. For nearly all tested cases, dynamic S-VNS outperforms the MSA and therefore seems to be the method of choice.

Furthermore, some of the main conceptual design aspects of the original design of the multiple plan and multiple scenario approach [40] should be modified in order to obtain the best possible results for the problem at hand. Instead of using a consensus function to determine the used current incumbent solution out of the set of solutions, the best solution out of the set should be used for this purpose. This may be caused by the fact that the problem we study uses a completely different objective function than the one used for the VRPTW by Bent and Van Hentenryck [40] (minimization of the number of rejected requests). It could be an interesting topic for further research to study the possible relation between the used objective function and the method used to determine the current incumbent solution. Additionally, the used search component does in fact have a strong influence on solution quality. More precisely, the used long term memory component induces an additional source of diversification when compared to our single solution based VNS approach. In combination with a strong set of shaking operators, this seems to drive the search process towards a random search like behavior, thus reducing solution quality. Nevertheless, the multiple scenario approach can be a powerful and robust stochastic solution method if the right design decisions are made.

Summing up, we can say that we were able to prove that integrating stochastic information about a relatively small portion of future requests (only return transports) into the process of planning vehicle routes for the DSDARP can be beneficial to solution quality. In our opinion, S-VNS clearly is the method of choice for this problem type. However, it seems that the MSA is the more robust method with respect to R and N, which in some cases might be favorable.

4 Stochastic time-dependent travel speeds

4.1 Introduction and related work

Based on the version of the dynamic dial-a-ride problem presented in Chapter 2, we present a second extension as a further step towards a more realistic representation of the problem faced by the Austrian ambulance service providers. For this purpose, we extend the DDARP by introducing stochastic time-dependent travel speeds. We denote the resulting problem as dynamic dial-a-ride problem with stochastic time-dependent travel speeds (DDARPTD). Note that, in this chapter, the return transports studied in Chapter 3 are being treated as purely dynamic requests again (as in the basic DDARP). This chapter is based on a working paper by Schilde et al. [18] and most passages are taken from there.

Most of the published articles related to vehicle routing problems assume travel speeds to be constant over time (e.g., for the dial-a-ride problem, see [16, 9, 14]). In reality, however, travel speeds are not constant at all. They are heavily influenced by many factors like, for example, traffic congestion caused by rush-hours, accidents, construction sites or just bad weather conditions. An example for this influence can be seen in Figure 4.1. It shows the average travel speed observed on a specific road segment in the city of Vienna depending on the time of day. The morning and afternoon peaks (rush hours) which are typical for inner-city roads are clearly visible. Also, the real (stochastic) travel speed observed during one specific day on the same road segment is shown. As can be seen, travel speeds are highly sensitive to the time of day and furthermore show significant stochastic fluctuations caused by different effects. Therefore, assuming travel speeds to be non-stochastic or even time-independent often causes planned schedules to fail with respect to time windows or ride time limitations.

Recent publications try to avoid this shortcoming by treating travel speeds as time-dependent. More precisely, in most of the publications it is assumed that a day can be divided into discrete time intervals which imply a characteristic average travel speed for each road inside a road network (see, for example, [47, 48, 20, 49, 50, 51]). Still, most of these approaches assume travel speeds to be deterministic. This means, that the travel speed in terms of average values for each interval is known a priori and is not influenced by any stochastic effects. Some authors (see, e.g., [52, 53, 54]) use a different approach to this problem. The idea is to incorporate time-dependent travel speeds in the process of calculating shortest paths. However, to the best of our

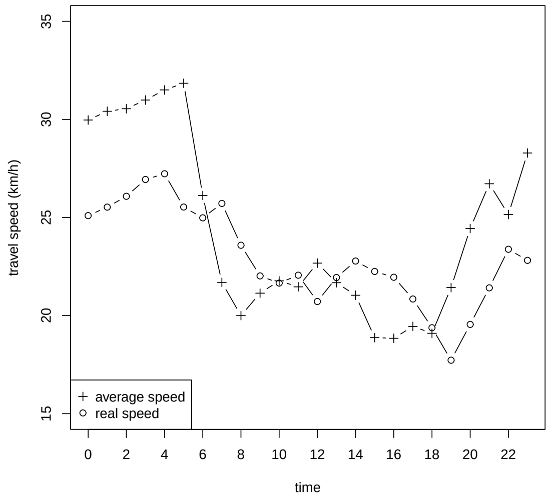

Figure 4.1: Time-dependent travel speeds over 24 hours: average versus real (stochastic) velocities.

knowledge, the algorithms used by these authors to create the actual vehicle routes do not take any stochastic information about future travel speeds into account in order to obtain a better solution. Instead, these methods still treat travel times as given and are restricted to reacting on changes in travel speeds.

As can be seen in Figure 4.1, the real travel speed during one day can, however, significantly deviate from time-dependent average values. Therefore, travel speeds should rather be treated as being stochastic in order to represent reality more precisely. This way, the reliability and productivity of the planned schedules can be improved significantly (see, e.g., [55, 56]). Especially in the case of conveying passengers, missed time windows and excessive ride times have a strong negative effect on the quality of service. This is even more important if the transported passengers are patients or elderly people. Furthermore, contrary to what could be expected, treating travel speeds as time-dependent and non-stochastic in an environment in which travel speeds in fact are stochastic does not necessarily lead to significant improvements on solution quality (see Section 4.4.2 for details).

In this chapter we assume time-dependent average values of travel speeds to be known from historical floating car data (FCD). In detail, we divide a day into 24 intervals, each implying a specific (known) average travel speed for each road segment in a real world road network. The corresponding FCD was collected and analyzed during a project in the city of Vienna (c.f., [57, 58, 59, 60]). This means that it is a realistic representation of the real world traffic situation during the observation interval including temporal as well as geographical correlations between travel speeds on different streets. Based on the analysis of this data, we know that real travel speeds regularly deviate from an interval's average travel speed by as much as 20% in both directions due to stochastic influences (e.g., accidents). In what follows, we address the question if stochastic algorithms (i.e., which take into account the available information about these stochastic deviations) can produce better solutions than myopic ones (i.e., which rely only on the fixed mean travel speeds for each interval or even only on constant average travel speeds). As the DDARPTD is a logical generalization of the DARP, it also belongs to the class of NP-hard combinatorial optimization problems. Hence, we adapt the four metaheuristic solution methods presented in Chapter 3 to the requirements of the problem at hand.

The remainder of this chapter is organized as follows. In Section 4.2, a detailed problem description is provided, followed by an overview on the used methods in Section 4.3. In Section 4.4 the used test instances and the corresponding computational results are explained, respectively. The chapter concludes with a summary in Section 4.5.

4.2 Problem definition

The dynamic dial-a-ride problem with stochastic time-dependent travel speeds is an extension to the basic DDARP presented in Chapter 2. It is based on a (directed) real world road network with stochastic time-dependent travel speeds. This means, we assume that going from node A to any other node B takes $\hat{T}(t, A, B) = \hat{T}_{avg}(t, A, B) + T_{stoc}(t, A, B)$ time units. Hereby, $\hat{T}_{avg}(t, A, B)$ is the time required to travel from A to B when leaving A at a given time t. This travel time is based on the average vehicle speeds during the affected time intervals and is assumed to be known a priori. The term $T_{stoc}(t, A, B)$ represents the stochastic influence on this travel time which is revealed only upon departure from A. Note, that we denote by $\check{T}(t, A, B)$ the time required to travel from point A to point B when the arrival time at point B should be t.

In the case of the DDARPTD, we need to slightly modify the maximum detour constraint due to the time-dependent nature of the problem. Let the time required to go directly from p_r to d_r departing at time t be $\hat{T}_{direct}(t, p_r, d_r)$. Let the time between the planned end of service at p_r and the planned start of service at d_r be T_{real}. Then, the equation $T_{real} \leq \hat{T}_{direct}(t, p_r, d_r) + 30$ should not be violated. However, due to the stochastic influence on travel speeds, we cannot guarantee that this equation will strictly hold. In case a travel time happens to be longer in reality than in the plan, a detour might end up being longer than 30 minutes. To penalize this effect, the amount of time by which a solution violates this maximum detour constraint is denoted as ride time violation. Furthermore, waiting times may only be planned when a vehicle is empty.

The top priority while planning the vehicle routes is again to minimize passenger dissatisfaction. Additionally, total costs have to be kept as low as possible. We therefore adapt our lexicographic objective function. The primary objective is to minimize the sum of tardiness, earliness and ride time violations over all routes. The secondary objective is the number of routes (vehicles used). The third objective is the total route duration. In general, solutions are compared according to the primary objective. In case two solutions are equal in terms of the primary objective, the secondary one is used for comparison. Only if both, primary and secondary objective, are equal for two solutions, comparison is based on the third objective.

4.3 Solution methods

The research described in this chapter aims at studying the effect of exploiting stochastic information about the future traffic situation while planning vehicle routes for the Austrian ambulance service providers. The question is again, if stochastic solution algorithms can obtain solutions of higher quality than myopic methods that ignore any stochastic information. For this purpose we adapt the metaheuristic solution methods presented in Chapter 3 in order to handle the stochastic time-dependent travel speeds in this problem.

The first method we adapt to the requirements of the DDARPTD with expected return transports is our dynamic variable neighborhood search approach presented in Section 3.3.3. This approach ignores any available stochastic information about future travel speeds and uses only the known time-dependent average travel speeds for planning.

This dynamic VNS method is the basis for our adaptation of the dynamic S-VNS approach presented in Section 3.3.4. Dynamic S-VNS is again the stochastic counterpart to dynamic VNS, taking stochastic information about future travel speeds into account while designing the vehicle routes for the Austrian ambulance service providers. The stochastic information is used to evaluate solutions based on sampled future travel speeds whenever two solutions need to be compared.

As a second pair we adapt the myopic multiple plan approach and the multiple scenario approach presented in Sections 3.3.5 and 3.3.6. We use the adapted version of our dynamic VNS as a local search component within the MPA and the MSA framework. The MSA requires some more adaptation effort than dynamic S-VNS in order to exploit the stochastic information about future traffic situations effectively. This is caused by the fact that the original concept of the MSA is based on the idea of generating gaps in the found solutions by removing sampled information from them after the search phase. This, however, does not work well for travel speeds, as removing them from a solution means to re-schedule them with known travel speeds. Thus, no information about the used sample (i.e., gaps) are left behind. The details for all required adaptations are presented in the following sections.

4.3.1 Simulation framework

As the DDARPTD is also a dynamic problem, we re-use the simulation framework developed for the DSDARP (see Section 3.3.1) as basis for our simulation environ-

Algorithm 9 Guaranteeing the FIFO property

```
1: $t \leftarrow t_0$
2: $d \leftarrow d_{ij}$
3: $t' \leftarrow t + \frac{d}{v_{ijk}}$
4: while $(t' > \bar{t_k})$ do
5:     $d \leftarrow d - v_{ijk}(\bar{t_k} - t)$
6:     $t \leftarrow \bar{t_k}$
7:     $k \leftarrow k + 1$
8:     $t' \leftarrow t + \frac{d}{v_{ijk}}$
9: end while
10: return $(t' - t_0)$
```

ment. It is already designed to continuously keep track of all transportation requests' status during execution, provides information about incoming new requests to our solution methods whenever necessary and manages simulation time.

We enhance this framework such that it is capable of managing stochastic time-dependent travel speeds. To be precise, we add an interface which allows the solver modules to request expected travel times between two locations for a given departure or arrival time. In this way the framework provides travel times based on the average speeds within the affected intervals. Only when simulation time advances, travel times including the actual stochastic influences are revealed to the solver modules. This means that the framework is based on the assumption that true (stochastic) travel times are revealed to a solver the moment a vehicle departs towards it's next stop. Thus, no updating of the travel times is performed while traveling the corresponding path. Also, the actual path is not re-calculated but only the corresponding travel time required for this path is being updated upon departure.

An important aspect in this extended framework is to guarantee the "first-in-first-out" property (FIFO [51], also known as "non-passing property", [61]). Besides the logical implications of this property, this also has systemic reasons. Our scheduling algorithm is based on a strategy that requires being able to find a definite departure time for any given arrival time (see the definition of $\check{T}$ in Section 4.2). In such situations, guaranteeing the FIFO property is crucial as otherwise multiple departure times could result in the same arrival time. This problem and its solution by guaranteeing the FIFO property are shown in Figures 4.2 and 4.3, respectively. We guarantee the FIFO property by using the approach presented by Ichoua et al. [51] as listed in Algorithm 9. This approach can easily be inverted such that the travel time is being calculated given a specific arrival time. We denote by t_0 the departure time at location i, by d_{ij} the total distance to be traveled, by v_{ijk} the velocity on link (i, j) during interval k, and by $\bar{t_k}$ the end of interval k.

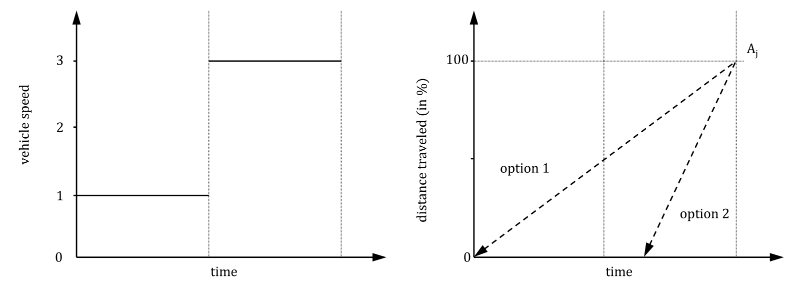

Figure 4.2: The problem arising without guaranteeing the FIFO property.

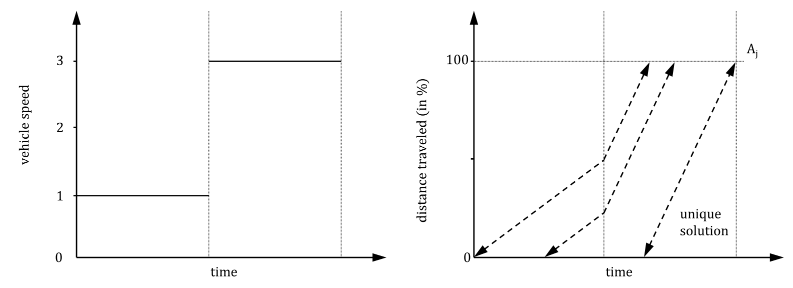

Figure 4.3: The solution when guaranteeing the FIFO property.

4.3.2 Scheduling algorithm

Our solution approaches are based on two phases. First, the sequence of transportation requests inside each route is determined. Second, the feasibility and timing for each route is determined using a scheduling algorithm. For this purpose, we present an alternative scheduling algorithm for the DDARPTD in this chapter which we call block scheduling algorithm (BSA). It is inspired by two ideas presented in the literature: the concept of scheduling blocks presented by Fu [56] (which is somewhat similar to the concept of zero split points presented by Parragh et al. [9]) and the forward time slack concept proposed by Savelsbergh [62]. A service block is defined as a sequence of nodes inside a route that starts and ends with the vehicle being empty. The forward time slack is originally defined as the maximum amount of time by which the departure from a node can be delayed without causing the route to be infeasible. For our scheduling algorithm, we define the forward time slack as the maximum amount of time by which the start of service at a node can be delayed without increasing tardiness with respect to its time window.

We decide not to adapt any of the two most commonly used scheduling algorithms proposed for the deterministic DARP [10, 43] (we use the second one for the DSDARP presented in Chapter 3) to the requirements of the DDARPTD because of the following reason. Shifting services with respect to the time they are performed

Algorithm 10 Block scheduling algorithm: Outline

```
for all stops along the route do
    Determine timing without waiting time
    if vehicle is empty before this stop then
        Add waiting time before this stop if needed
        Remember new block characteristics
    else
        Update block characteristics
    end if
end for
for all blocks in backwards order do
    if block can and should be shifted then
        Shift block by adding waiting time in front
        Update block characteristics
        while shift caused problems that cannot be ignored do
            Undo part of the shift to correct problems
            Update block characteristics
        end while
    end if
end for
```

can have a direct influence on all following travel times. Additionally, the problem at hand includes only soft time windows, soft maximum ride time constraints (due to the stochastic influences on travel speeds) and no waiting time is allowed whenever a person is aboard a vehicle. Therefore, extensive modifications to the existing scheduling algorithms would be required, whereas a new concept can be designed especially for this setting and thus seems to be more appropriate.

Unlike the two mentioned scheduling algorithms [10, 43], the BSA corrects time window violations by shifting complete scheduling blocks instead of individual services (as described in detail in the following paragraphs). If waiting time with patients aboard is allowed, this might lead to schedules that have a higher tardiness with respect to the time windows than the ones obtained by the other methods. However, if speed is not a crucial aspect, this could be overcome by adding a third phase that performs an additional service-wise shift after the block-wise shift. The major advantage of BSA compared to the other two scheduling methods is the fact that BSA is specifically designed to handle time-dependent travel speeds while determining the schedule for a route. This design guarantees that no infeasibility can be caused by the interaction between a shifting operation and the changing travel speeds. In other words, the existing algorithms can cause schedules in which vehicles are planned to arrive after the next stop's beginning of service. This is due to the fact that shifting a stop's service into the future can have a direct influence on all following travel times (as travel speeds are time-dependent).

Algorithm 11 Block scheduling algorithm: Phase 1

```
1: $i \leftarrow 0$
2: $Q_0 \leftarrow 0$
3: for $n = \{1, \dots, |S|\}$ do
4:     $A_n \leftarrow D_{n-1} + \hat{T}(D_{n-1}, n-1, n)$
5:     $B_n \leftarrow A_n$
6:     $D_n \leftarrow B_n + d_n$
7:     $\mathcal{Q}_n \leftarrow \mathcal{Q}_{n-1} + q_n$
8:     if $\mathcal{Q}_n > Q$ then
9:         return Infeasible
10:    end if
11:    if $(Q_n = 0) \wedge (A_n < e_n)$ then
12:        $A_n \leftarrow e_n$
13:        $D_{n-1} \leftarrow A_n - \check{T}(A_n, n-1, n)$
14:        $B_n \leftarrow A_n$
15:        $D_n \leftarrow B_n + d_n$
16:        $i \leftarrow i + 1$
17:        $\Theta_i^{start} \leftarrow n$
18:        $\Theta_i^{waiting} \leftarrow D_{n-1} - B_{n-1} - d_{n-1}$
19:        $\Theta_i^{earliness} \leftarrow \max\{e_n - B_n, 0\}$
20:        $\Theta_i^{slack} \leftarrow \max\{l_n - B_n, 0\}$
21:    else
22:        $\Theta_i^{earliness} \leftarrow \max\{\max\{e_n - B_n, 0\}, \Theta_i^{earliness}\}$
23:        $\Theta_i^{slack} \leftarrow \min\{\max\{l_n - B_n, 0\}, \Theta_i^{slack}\}$
24:    end if
25: end for
```

The BSA is a two stage method as outlined in Algorithm 10. In the first stage, the algorithm creates an initial schedule for the given route by using forward propagation. Hereby waiting times are permitted only directly before departing towards a pickup service if the vehicle is empty (not after arriving at the pickup location). Such a point along the route also indicates the beginning (and end) of a service block. During this phase, all required parameters of the respective service blocks are stored for the second phase. This initial schedule is then refined in phase 2 by introducing additional waiting time before service blocks in order to reduce earliness with respect to time windows. This second phase steps through the found blocks in backwards order and introduces just enough waiting time in front of the first stop of each block such that earliness is minimized without increasing tardiness. Due to the time-dependent nature of travel speeds, this can cause problems if the changing travel times lead to negative waiting time behind the current block. This effect is then compensated by iteratively removing part of the inserted waiting time again.

A detailed outline is given in Algorithms 11 and 12. We use the following notation:

Algorithm 12 Block scheduling algorithm: Phase 2

```
1: for $i = \{|\Theta| - 1, \dots, 1\}$ do
2:   if $(\Theta_i^{slack} > 0) \wedge (\Theta_i^{earliness} > 0) \wedge (\Theta_{i+1}^{waiting} > 0)$ then
3:     $s \leftarrow \min\{\Theta_i^{slack}, \Theta_i^{earliness}, \Theta_{i+1}^{waiting}\}$
4:     $B_{orig} \leftarrow B_{\Theta_{i+1}^{start}-1}$
5:     $\Theta_i^{waiting} \leftarrow \Theta_i^{waiting} + s$
6:     $A_{\Theta_i^{start}} \leftarrow A_{\Theta_i^{start}} + s$
7:     $B_{\Theta_i^{start}} \leftarrow A_{\Theta_i^{start}}$
8:     $D_{\Theta_i^{start}-1} \leftarrow A_{\Theta_i^{start}} - \check{T}(A_{\Theta_i^{start}}, \Theta_i^{start} - 1, \Theta_i^{start})$
9:     $\Theta_i^{tardiness} \leftarrow 0$
10:    $\phi \leftarrow 0$
11:    for $n = \{\Theta_i^{start} + 1, \dots, \Theta_{i+1}^{start} - 1\}$ do
12:      $D_{n-1} \leftarrow B_{n-1} + d_{n-1}$
13:      $A_n \leftarrow D_{n-1} + \hat{T}(D_{n-1}, n-1, n)$
14:      $B_n \leftarrow A_n$
15:      $\Theta_i^{tardiness} \leftarrow \max\{\max\{B_n - l_n, 0\}, \Theta_i^{tardiness}\}$
16:      $\Theta_i^{slack} \leftarrow \min\{\max\{l_n - B_n, 0\}, \Theta_i^{slack}\}$
17:      if $n = \Theta_{i+1}^{start} - 1$ then
18:        $\Theta_{i+1}^{waiting} \leftarrow D_n - B_n - d_n$
19:        if $((\phi < \phi_{max}) \wedge (\Theta_i^{tardiness} > 0)) \vee (\Theta_{i+1}^{waiting} < 0)$ then
20:          $\phi \leftarrow \phi + 1$
21:          $s_{corr} \leftarrow ((B_n - B_{orig})/s)(-\min\{-\Theta_i^{tardiness}, \Theta_{i+1}^{waiting}\})$
22:          $s \leftarrow s - s_{corr}$
23:          $\Theta_i^{waiting} \leftarrow \Theta_i^{waiting} - s_{corr}$
24:          $A_{\Theta_i^{start}} \leftarrow A_{\Theta_i^{start}} - s_{corr}$
25:          $B_{\Theta_i^{start}} \leftarrow A_{\Theta_i^{start}}$
26:          $D_{\Theta_i^{start}-1} \leftarrow A_{\Theta_i^{start}} - \check{T}(A_{\Theta_i^{start}}, \Theta_i^{start} - 1, \Theta_i^{start})$
27:          $n \leftarrow \Theta_i^{start} + 1$
28:          $\Theta_i^{tardiness} \leftarrow 0$
29:        end if
30:      end if
31:    end for
32:  end if
33: end for
```

S is the set of nodes to be visited by the current route (in chronological order, 0 and $|S|$ are the indices of the depot nodes), $\mathcal{Q}_n$ is the number of passengers aboard the vehicle when arriving at node n, Q is the vehicle capacity, A_n is the arrival time at node n, B_n is the beginning time of service at node n, D_n is the departure time from node n, d_n is the service time at node n and Θ_i is used to store information about service block i. The first stage consists of creating an initial schedule by using forward propagation (Algorithm 11). Hereby, waiting times (in order to reduce earliness with respect to time windows) are only permitted whenever no person is aboard the vehicle (Lines 12 to 15). As all time windows and the maximum ride time limitation are soft, the only definitive infeasibility would be a violation of the vehicle capacity (Line 9). This way we obtain a feasible schedule consisting of a sequence of service blocks connected by waiting time and subsequent deadheading movement. During this first phase, we keep track of four different properties of each block (Lines 17 to 23): its starting index (Θ_i^{start}), the maximum earliness within the block ($\Theta_i^{earliness}$), the minimum forward time slack within the block (Θ_i^{slack}) and the waiting time before the block ($\Theta_i^{waiting}$).

In the second phase, the algorithm steps backwards through the schedule blocks and postpones complete blocks in order to reduce earliness (Algorithm 12). More precisely, the waiting time before each block i is increased by the minimum out of $\Theta_i^{earliness}$, Θ_i^{slack} and $\Theta_{i+1}^{waiting}$ (Lines 3 to 8). Then, all timings inside the block and the block's properties are updated accordingly using forward propagation again (Lines 12 to 16). This way, we obtain a schedule which minimizes earliness and tardiness with respect to time windows without waiting time within a service block. However, due to the time-dependent nature of travel speeds, postponing a service block like this can lead to negative waiting time after the block (i.e., we would need to shift the subsequent block - which we already shifted in a previous iteration) or an increase in tardiness inside the block. This is caused by the fact that vehicle movements might be shifted into later time intervals with lower travel speeds. The BSA iteratively compensates this effect before continuing with the preceding block. For this purpose it determines the amount of negative waiting time caused by the current block movement relative to the performed postponement. Then, it undoes part of the block's postponement accordingly (Lines 21 to 28). This correction is performed iteratively as long as the waiting time after the block is negative or up to ϕ_{max} times in order to compensate for an increase in tardiness inside the block caused by the shift. This limitation is used to avoid excessive computation time.

4.3.3 Dynamic variable neighborhood search

The first solution method we adapt to the requirements of the DDARPTD is our dynamic VNS presented for the DSDARP in Section 3.3.3. An outline of the adapted method is given in Algorithm 13.

The method is adapted to take time-dependent travel speeds into account while planning. These travel speeds are provided by the simulation framework for a specific departure or arrival time and are based on the average vehicle speeds within each affected interval. We assume the stochastic travel speed for any path inside the road network to be known with certainty the moment a vehicle starts traveling that path. Travel speeds are not updated while traveling a path nor is the path itself re-calculated depending on the current traffic situation. This new information then has to be incorporated into the current solution as well. Therefore, dynamic

Algorithm 13 Structure of dynamic VNS for the DDARPTD

```
1: $x \leftarrow$ InitialSolution()
2: $\mathcal{N}(\kappa) \leftarrow$ SelectFirstNeighborhood()
3: while StoppingCriterionNotMet() do
4:     $x \leftarrow$ RescheduleWithTrueTravelSpeeds($x$)
5:     $x \leftarrow$ InsertNewRequests($x$)
6:     $x' \leftarrow$ ShakeSolution($x, \mathcal{N}(\kappa)$)
7:     $x \leftarrow$ MoveOrNot($x, x'$)
8:     $\mathcal{N}(\kappa) \leftarrow$ SelectNextNeighborhood($\kappa$)
9: end while
10: return $x$ as best found solution
```

VNS needs to re-schedule all subsequent requests in an affected route according to this real travel time whenever a vehicle departs towards it's next stop (Line 4). By making the mentioned assumption, this process is facilitated by enabling the algorithm to determine a specific (real) arrival time the moment a vehicle departs from it's current position. We are confident that this simplifying assumption still represents a sufficiently accurate model of reality. We base this on the observation that travel times within a city tend to be rather short and thus traffic situations can be assumed to stay roughly unchanged during this time. The remainder of this adapted dynamic VNS stays the same as the original dynamic VNS.

As in the original version of dynamic VNS (see Section 3.3.3) we use a set of four different neighborhood operators based on the ones proposed by Parragh et al. [9] during the shaking phase of our dynamic VNS algorithm (Line 6). Each of these operators is again used in five different intensity levels $\kappa = \{1 \dots 5\}$. The first five neighborhoods use a move operator. This operator randomly removes κ transportation requests from a randomly selected route and re-inserts them into any route at the position where they fit best. By this we mean that this position causes the smallest possible deterioration in solution quality. The second set of five neighborhood operators uses a swap operator. It randomly selects two routes and removes up to κ consecutive requests from each of them starting at a randomly selected position, respectively. The removed requests are then re-inserted into the corresponding other route at the position where they fit best. The third set uses a chain operator. It starts by randomly selecting an origin route and a destination route. After that, it removes a sequence of up to κ consecutive requests from the origin route and re-inserts them into the destination route where they fit best. It then iterates κ times using the destination route as new origin route and a randomly selected new destination route. The last neighborhood set is based on a zero split operator. It randomly selects one route and determines all positions inside the route at which the corresponding vehicle is empty ("zero split points"). It then randomly selects two of these points, whereby up to $\kappa - 1$ other such points may be in between these two points. Finally, it removes all transportation requests between the selected points

Algorithm 14 Structure of dynamic S-VNS for the DDARPTD

```
1: $x \leftarrow$ InitialSolution()
2: $\mathcal{N}(\kappa) \leftarrow$ SelectFirstNeighborhood()
3: $Z \leftarrow$ SampleFutureTravelSpeeds()
4: $x \leftarrow$ Reschedule($x, Z$)
5: $u, i \leftarrow 0$
6: while StoppingCriterionNotMet() do
7:     $i \leftarrow i + 1$
8:     $x \leftarrow$ RescheduleWithTrueTravelSpeeds($x$)
9:     $x \leftarrow$ InsertNewRequests($x$)
10:    $x' \leftarrow$ ShakeSolution($x, \mathcal{N}(\kappa)$)
11:    if $i - u > U$ then
12:        $u \leftarrow i$
13:        $Z \leftarrow$ SampleFutureTravelSpeeds()
14:        $x \leftarrow$ Reschedule($x, Z$)
15:        $x' \leftarrow$ Reschedule($x', Z$)
16:    end if
17:    $x \leftarrow$ MoveOrNot($x, x'$)
18:    $\mathcal{N}(\kappa) \leftarrow$ SelectNextNeighborhood($\kappa$)
19: end while
20: return $x$ as best found solution
```

and re-inserts these requests into any route at the position where they fit best.

We start our search using the move neighborhood with intensity level $\kappa = 1$. For each neighborhood we randomly create one neighboring solution. If this solution is not better than the current incumbent solution, we continue with intensity level $\kappa = \kappa + 1$, otherwise we continue with the first operator and $\kappa = 1$. When the maximum value for the intensity is reached, we switch to the next neighborhood operator (move $\rightarrow$ swap $\rightarrow$ chain $\rightarrow$ zero split) with intensity level $\kappa = 1$. If the maximum intensity is reached for the last neighborhood operator, we switch back to the move neighborhood (Line 8).

4.3.4 Dynamic stochastic variable neighborhood search

The second algorithm we apply to this variant of the dynamic dial-a-ride problem is an adaptation of the dynamic stochastic variable neighborhood search approach (see Section 3.3.4). The main idea behind the S-VNS concept stays the same: determine the quality of any given solution based on samples of future developments. The outline of this adaptation is given in Algorithm 14.

Most of the overall structure of dynamic S-VNS is not changed in this adaptation. As dynamic S-VNS is meant to take stochastic information about future travel speeds into consideration while planning, the current incumbent solution is evaluated based on a sample of future travel speeds. Therefore, the algorithm determines possible future deviations from each road segment's average travel speed for each future interval by sampling the corresponding distribution parameters (Lines 3 and 4). At the beginning of each iteration the algorithm adapts the current incumbent solution to stochastic travel speeds if they are known by now (Line 8). It then inserts all new requests which have become known during the last iteration into this solution (Line 9). This is followed by a shaking step using the same neighborhood structures as previously described for dynamic VNS (Line 10). After every U iterations, the algorithm updates the used sample and re-schedules both x and x' accordingly (Lines 11 to 16). Then, the solution which has the better sample average estimator with respect to the current sample is selected as new current incumbent solution (Line 17). Due to the computationally rather expensive sampling of future travel speeds we again use a single sample for this evaluation. In the original S-VNS concept (see [38]) the sampled information is only used for this comparison and not integrated into the current incumbent solution before continuing. As it turns out, this doesn't make much sense for sampled travel speeds as the current incumbent solution would be scheduled with average (time-dependent) travel speeds and doesn't include any viable information about the used sample. Therefore, in a setting like this it is essential to keep the sampled information included in the solution that is to be executed in order to obtain significant benefits over an algorithm that uses only average travel speeds for planning.

4.3.5 Multiple plan approach

As a myopic base for the second pair of metaheuristics the multiple plan approach is adapted to the requirements of the DDARPTD. All modifications are based on the adaptation for the DSDARP (see Section 3.3.5) and introduce all changes required to cope with stochastic time-dependent travel speeds. As underlying search procedure we use our dynamic VNS in the version adapted to the DDARPTD. This way, we again guarantee all differences found between the results of our four methods to be caused by the essential conceptual design (myopic versus stochastic, long term memory versus no long term memory) and not by the underlying search method.

An outline of the adapted multiple plan approach is shown in Algorithm 15. The main idea behind the MPA is to use a pool of solutions as long term memory in which each unique solution found during the search process is stored is kept unchanged in this modification (Line 15). At every point in time, the algorithm uses one of these solutions as a current incumbent solution (which is to be executed by the vehicles; Line 5). To guarantee the feasibility of all solutions in the pool, solutions which are incompatible with decisions made in the current incumbent solution are eliminated from the pool whenever necessary (Line 8). This way, the resulting solution is defined by the sequence of actions taken during execution.

According to Bent and Van Hentenryck [40], the best strategy to select the current incumbent solution is based on a consensus function similar to a least commitment strategy. This means that the solution most similar to all other solutions in the long

Algorithm 15 Structure of the MPA for the DDARPTD

```
1: $P \leftarrow$ InitialSolution()
2: $\mathcal{N}(\kappa) \leftarrow$ SelectFirstNeighborhood()
3: **while** StoppingCriterionNotMet() **do**
4:    $\bar{x} \leftarrow$ RescheduleWithTrueTravelSpeeds($\bar{x}$) $\forall \bar{x} \in P$
5:    $x \leftarrow$ SelectCurrentIncumbent($P$)
6:    **for** $\bar{x} \in P$ **do**
7:       **if** $(\bar{x} \neq x) \wedge$ (Timeout($\bar{x}, x$) $\vee$ Departure($\bar{x}, x$)) **then**
8:          $P \leftarrow P \setminus \{\bar{x}\}$
9:       **else**
10:         InsertNewRequests($\bar{x}$)
11:      **end if**
12:   **end for**
13:   $x' \leftarrow$ ShakeSolution($x, \mathcal{N}(\kappa)$)
14:   **if** $x' \notin P$ **then**
15:      $P \leftarrow P \cup \{x'\}$
16:   **end if**
17:   $\mathcal{N}(\kappa) \leftarrow$ SelectNextNeighborhood($\kappa$)
18: **end while**
```

term memory is selected. Previous findings for the DSDARP (see Chapter 3) indicate that this strategy, however, may not be the best choice under all circumstances. As the DDARPTD is structurally similar to the DSDARP, we decide to use the best solution in the pool as our current incumbent solution as we do for the DSDARP.

4.3.6 Multiple scenario approach

Based on the MPA, we modify the multiple scenario approach in order to exploit the available stochastic information about future travel speeds. All adaptations are based on the modified version of the MSA presented for the DSDARP (see Section 3.3.6). An outline of this modification is shown in Algorithm 16.

The main difference between the MPA and the MSA is that the latter incorporates stochastic information in the search process, while the former does not. For this purpose, the multiple scenario approach samples future travel speed deviations the same way as dynamic S-VNS. Contrary to S-VNS, it does not only use this information for comparing two solutions by means of sample average estimators, but uses this information during the search process. To be precise, it re-schedules the current incumbent solution using the sampled stochastic deviations before the search process (Lines 4 and 8 to 12). As for S-VNS, the used sample is updated every U iterations. This way, the search process is intended to be guided towards solutions which are of high quality regarding the used sample in order to increase the diversity of solutions in the long term memory. Note, that it is again (as for S-VNS) crucial to keep the stochastic information (i.e., the current sample of future

Algorithm 16 Structure of the MSA for the DDARPTD

```
1: $P \leftarrow$ InitialSolution()
2: $\mathcal{N}(\kappa) \leftarrow$ SelectFirstNeighborhood()
3: $Z \leftarrow$ SampleFutureTravelSpeeds()
4: $\bar{x} \leftarrow$ Reschedule($\bar{x}, Z$) $\forall \bar{x} \in P$
5: $u, i \leftarrow 0$
6: **while** StoppingCriterionNotMet() **do**
7:    $i \leftarrow i + 1$
8:    **if** $i - u > U$ **then**
9:       $u \leftarrow i$
10:      $Z \leftarrow$ SampleFutureTravelSpeeds()
11:      $\bar{x} \leftarrow$ Reschedule($\bar{x}, Z$) $\forall \bar{x} \in P$
12:   **end if**
13:   $\bar{x} \leftarrow$ RescheduleWithTrueTravelSpeeds($\bar{x}$) $\forall \bar{x} \in P$
14:   $x \leftarrow$ SelectCurrentIncumbent($P$)
15:   **for** $\bar{x} \in P$ **do**
16:      **if** $(\bar{x} \neq x) \wedge$ (Timeout($\bar{x}, x$) $\vee$ Departure($\bar{x}, x$)) **then**
17:         $P \leftarrow P \setminus \{\bar{x}\}$
18:      **else**
19:         InsertNewRequests($\bar{x}$)
20:      **end if**
21:   **end for**
22:   $x' \leftarrow$ ShakeSolution($x, \mathcal{N}(\kappa)$)
23:   **if** $x' \notin P$ **then**
24:      $P \leftarrow P \cup \{x'\}$
25:   **end if**
26:   $\mathcal{N}(\kappa) \leftarrow$ SelectNextNeighborhood($\kappa$)
27: **end while**
```

travel speeds) included in all the stored solutions instead of removing it before they are stored.

4.4 Computational experiments

In order to study the effect of exploiting the available stochastic information about future travel speeds while planning the vehicle routes for the Austrian ambulance service providers, we create a set of real world based test instances. The ride times in our test instances are based on historical data gathered during a recent floating car data (FCD) project in the city of Vienna. Additionally, the distribution parameters for the interarrival times of the requests serviced during this period described in the master thesis of Kritzinger [46] (see Section 3.4.1) are used again. All computational experiments are performed on one core of a SUN Fire X2270 server with 2 quad-

	$\Lambda = 0\%$				$\Lambda = 30\%$			
	inbound		outbound		inbound		outbound	
N	static	dynamic	(dynamic)	total	static	dynamic	(dynamic)	total
100%	113.2	0.0	111.6	224.8	81.2	32.0	111.6	224.8
200%	211.8	0.0	207.2	419.0	151.4	60.4	207.2	419.0
300%	309.8	0.0	303.0	612.8	216.0	93.8	303.0	612.8
400%	410.0	0.0	401.2	811.2	295.6	114.4	401.2	811.2

	$\Lambda = 50\%$				$\Lambda = 80\%$			
	inbound		outbound		inbound		outbound	
N	static	dynamic	(dynamic)	total	static	dynamic	(dynamic)	total
100%	58.0	55.2	111.6	224.8	23.0	90.2	111.6	224.8
200%	105.8	106.0	207.2	419.0	43.4	168.4	207.2	419.0
300%	152.8	157.0	303.0	612.8	59.2	250.6	303.0	612.8
400%	213.2	196.8	401.2	811.2	79.0	331.0	401.2	811.2

Table 4.1: Average number of transportation requests per instance class. (Λ - degree of dynamism, N - relative size of the instance set)

core Intel Xeon X5550 processors (2.66 GHz) with 24 GB of shared memory. The algorithms are implemented using C++ and the GNU compiler g++ in its version 4.1.2 on CentOS 5.5.

4.4.1 Test instances

We assume that a day consists of 24 time intervals, each of 1 hour length. For each link inside the used real world road network and each of the intervals we can therefore determine an average vehicle speed. These travel speeds are used as a planning basis by the myopic approaches (see Section 4.3). In addition, we generate stochastic deviations from these average velocities. This is done by drawing uniformly distributed random samples in the range $\Delta_i = [\underline{\Delta_i}, \overline{\Delta_i}]$ for each link inside the network and each interval i, whereby $\underline{\Delta_i} = \min\{\max\{\underline{B}_{abs}, \underline{B}_{rel}\}, \overline{B}_{abs}\}$ and $\overline{\Delta_i} = \max\{\min\{\overline{B}_{abs}, \overline{B}_{rel}\}, \underline{B}_{abs}\}$. Here, $\overline{B}_{abs} = 1.2$ and $\underline{B}_{abs} = 0.8$ represent the allowed upper and lower bound for deviations from the average velocity of interval i as a factor. For each interval $i > 0$, the terms $\overline{B}_{rel} = 1.1\Delta_{i-1}$ and $\underline{B}_{rel} = 0.9\Delta_{i-1}$ define the upper and lower bound for divergence from the previous interval's deviation Δ_{i-1}. For interval $i = 0$ these values are set to $\overline{B}_{rel} = \overline{B}_{abs}$ and $\underline{B}_{rel} = \underline{B}_{abs}$. This way, neighboring intervals' deviations are correlated in a way which we assume to be sufficiently realistic.

We create our transportation requests based on distribution parameters derived from real world data on daily operations of an Austrian ambulance service provider in the city Graz during one year. We hereby assume the parameters to be sufficiently representative for the city of Vienna as well. In addition to these parameters, we use a geocoded set of patient and hospital locations in the city of Vienna corresponding to the original locations serviced within one year. To model the geographical distribution of the generated requests, this set includes an occurrence counter for each location that is used to assign locations to created requests via roulette wheel. By sampling the distribution of the interarrival time of two consecutive requests we determine the arrival times for all requests. Similarly, the distribution of the time between an incoming request and the latest time of arrival at the hospital is sampled to construct the requests' time windows. By varying the distribution of interarrival times we create instances with $N = \{100\%, 200\%, 300\%, 400\%\}$ the number of requests arising in reality. A more detailed description of the process used to determine the distribution parameters can be found in Section 3.4.1.

For each of the 4 settings for N we created 5 test instances, each representing one working day of 10 hours length. To be precise, each instance consist of 50% inbound (home to hospital, static or dynamic) and 50% outbound (hospital to home, always dynamic) requests. Table 4.1 shows the average number of transportation requests that have to be serviced in each set of test instances. Additionally, the instances are created in a way such that the inbound requests can be selected to be static or dynamic by defining the desired degree of dynamism. For our computational experiments, we use degrees of dynamism of $\Lambda = \{0\%, 30\%, 50\%, 80\%\}$ to see how this factor influences our results.

The inbound requests are created by sampling the corresponding distributions using the modified parameters as follows. Starting at time $t = 0$, the arrival time a_1 of the first request is determined according to the corresponding distribution of the interarrival time. Iteratively, the occurrence time of the follow-up request a_i is determined based on the arrival time of the previous request a_{i-1}. We stop when a_i exceeds the considered period of 10 hours. For each of these transportation requests, the end of the delivery time window is determined by again sampling the corresponding distribution. The length of the delivery time windows is set to 30 minutes. The end of the 30 minute pickup time window is determined as the starting time of the delivery time window minus the time required to travel directly from pickup to delivery. The same procedure is used to generate the static transportation requests. The only difference is that, after creation, a_i is set to a value of 0 for each of the static requests.

Based on the resulting set of inbound requests, a set of outbound requests is generated. We assume that each inbound request, no matter whether static or dynamic, causes a corresponding outbound request. In this case, the arrival time a_r is sampled using the distribution for the time between the request's latest arrival time at the hospital and the beginning of the outbound request's pickup time window. The time window for pickup at the hospital starts at a_r and is set to 60 minutes. The start of the time window for arrival at the patient's home location is calculated using the average time required to travel directly from the hospital to the patient's home location. This time window has a length of 90 minutes, which is 60 minutes plus the maximum ride time allowed for the patient (30 minutes).

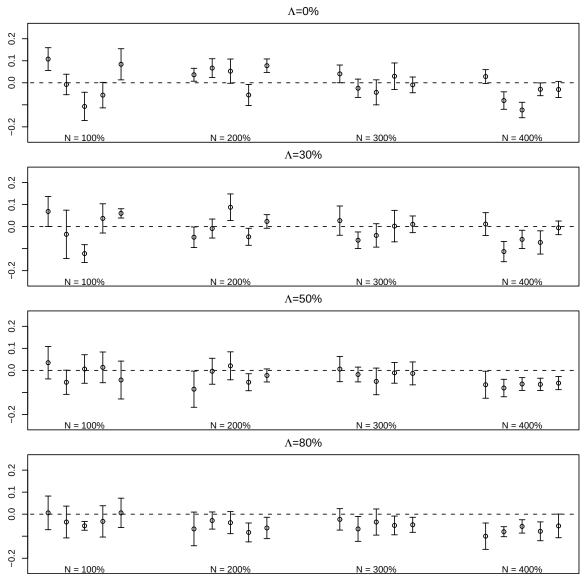

Figure 4.4: Average (circles) and 95% confidence intervals (whiskers) of the relative gaps in the primary objective obtained by the MPA with time-dependent travel speeds and the MPA with constant travel speeds (Λ - degree of dynamism, N - instance size).

Finally, we create a greedy solution for each of the test instances using the modified cheapest insertion procedure described in Section 3.3.2 under the assumption that all requests are known a priori. The number of vehicles used in this solution is increased by 10% and used as the maximum number of vehicles available to all four algorithms.

4.4.2 Results

In what follows, we present the results obtained during 5 independent runs on our test instances in terms of relative solution quality. This means, that results for the stochastic methods are given as percentage relative to the results obtained by their myopic counterparts. Also, results obtained with myopic approaches using time-dependent travel speeds are relative to the ones obtained when using constant travel speeds. A positive percentage indicates that the corresponding method yields a better solution quality than the approach it is compared to, whereas a negative value indicates the opposite. All results are obtained using an update frequency of $U = 100$ iterations for the samples of our stochastic methods.

At first, we test if using time-dependent travel speeds in our myopic approaches brings a benefit over using non-time-dependent (i.e., constant for the complete day) average travel speeds. We expect to observe significant improvements when using

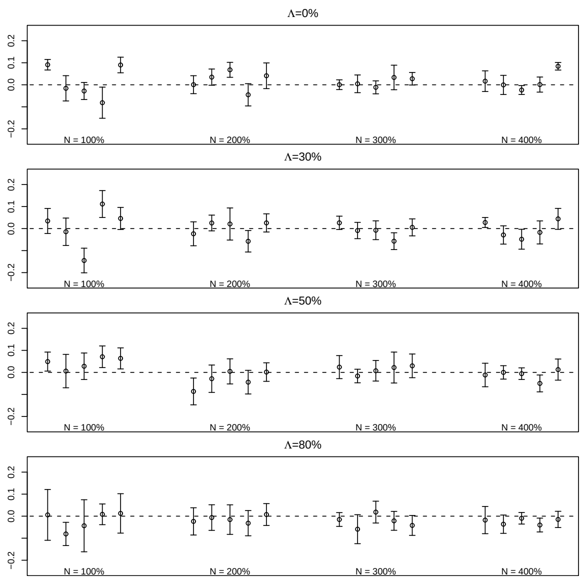

Figure 4.5: Average (circles) and 95% confidence intervals (whiskers) of the relative gaps in the primary objective obtained by dynamic VNS with time-dependent travel speeds and dynamic VNS with constant travel speeds (Λ - degree of dynamism, N - instance size).

time-dependent travel speeds. However, we find that for our setting with stochastic travel speeds, there is no reliable improvement when using time-dependent speeds compared to using constant speeds. This can be seen in Figure 4.4 for the MPA and Figure 4.5 for dynamic VNS. The figures show the average values (circles) and 95% confidence intervals for each of the five test instances and each of the different settings for N and Λ. A result above the 0% line indicates an improvement over the result obtained by the compared method whereas a result below this line indicates the opposite. These confidence intervals indicate that no stable benefit can be obtained by just using time-dependent travel speeds in the given setting with stochastic travel speeds. Especially for large problem sets ($N = 400\%$), even the opposite seems to be true.

The explanation for this finding is as follows. The main assumption when using time-dependent travel speeds is, that they are likely to be a better representation of the true travel speeds than is constant average speed. This, however, is not necessarily the case if there is a stochastic influence on true travel speeds. An example for the expected situation is given in Figure 4.6. As can be seen in the lower diagram (showing the absolute values of the deviations from true stochastic speeds), time-dependent speed is, on average, a better approximation of the true travel speed than constant average speed. The unexpected case is shown in Figure 4.7. Here, constant travel speeds are the better approximation on average. For our problem

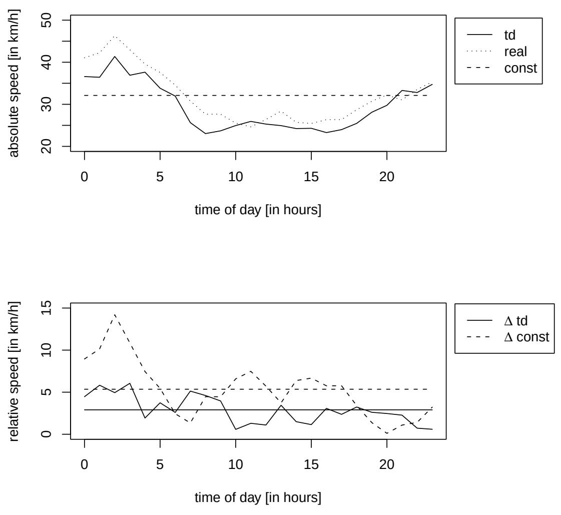

Figure 4.6: Comparison of the travel speeds showing the expected case (td - time-dependent speed, real - stochastic speed, const - constant average speed).

setting, both cases occur almost equally often thus leading to no reliable benefit from using time-dependent travel speeds while planning.

As mentioned earlier, the main focus of our research is to find out if our stochastic algorithms (dynamic S-VNS and the MSA) are able to obtain better solutions for this problem than our myopic methods (dynamic VNS and the MPA). A summary of all results obtained for $\Lambda = \{0\%, 30\%, 50\%, 80\%\}$ is shown in Tables 4.2, 4.3, 4.4 and 4.5, respectively. The columns titled gap_1 relate to the primary objective (sum of tardiness, earliness and ride time violations), the columns named gap_2 and gap_3 relate to the secondary (number of vehicles used) and tertiary (total route duration) objective functions, respectively. Note, that we use a lexicographic objective function. This means that only the primary objective function can be used directly to compare solution quality. Results for the secondary and tertiary objective function are presented for the sake of completeness only.

The first finding is that the conceptual design of the stochastic methods as proposed for the DSDARP (see Section 3.3) cannot be directly used for the problem at hand. The reason for this is that both algorithms are based on the same idea of producing gaps in the schedule of planned solutions which, at a later point in time, can be used to accommodate additional requests when they become known. The MSA does this by inserting and removing sampled future requests during the search process while dynamic S-VNS tends to prefer more robust solutions by using sample based comparison. This works well in the case of sampled transportation requests, but doesn't make much sense in the case of sampled travel speeds, as no information

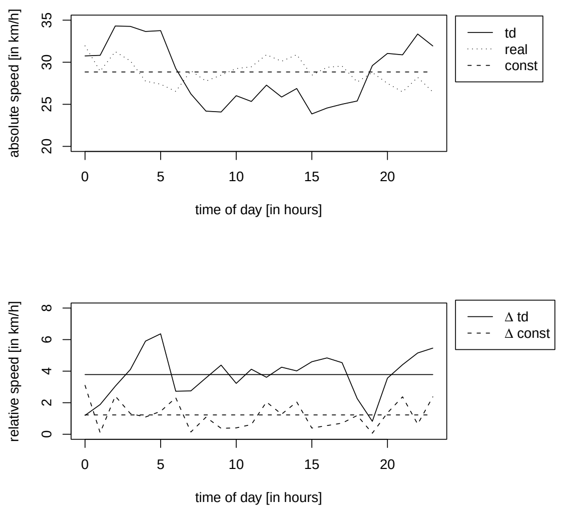

Figure 4.7: Comparison of the travel speeds showing the unexpected case (td - time-dependent speed, real - stochastic speed, const - constant average speed).

(in terms of gaps) is left after re-scheduling a solution with known travel speeds. Therefore, both methods have to be adapted in order to keep information about the sample included in their solutions (see Section 4.3). Still, this adaptation alone does not lead to the desired benefits.

Additionally, the used samples themselves need to show specific characteristics in order to achieve reasonably stable improvements in solution quality. Specifically, the samples of possible future travel speeds should only include negative effects. This means, that only during intervals for which the sample indicates a travel speed lower than the time-dependent average speed, the sample should be used for planning/evaluating. For all other intervals, the known time-dependent average speed should be used instead. In other words, in order to be beneficial, a sample should not represent a case that is better than the average situation. It thus seems that planning more conservatively (i.e., with travel times longer than average) has a positive effect on the obtained results.

In the static case ($\Lambda = 0\%$, Table 4.2), the MSA yields an average improvement of 12.25% compared to the solutions found by the MPA. Dynamic S-VNS achieves 11.30% average improvement over dynamic VNS in this case. This indicates, that (assuming that all inbound requests are known in advance whereas all outbound requests are dynamic) both stochastic methods can achieve reasonable improvements compared to the myopic methods by using the very limited amount of stochastic information that is available. In other words, taking stochastic deviations from time-dependent travel speeds into account while planning leads to significantly better

	gap_1 in %		*gap_2* in %		*gap_3* in %	
N	MSA	S-VNS	MSA	S-VNS	MSA	S-VNS
100%	10.63	**11.10**	1.21	-0.97	0.69	-0.42
200%	11.45	**11.53**	3.21	-1.76	3.01	-0.46
300%	**12.63**	10.87	4.29	-3.54	3.58	-2.44
400%	**14.31**	11.69	3.00	-1.55	2.97	-1.21
avg.	**12.25**	11.30	2.92	-1.96	2.56	-1.13

Table 4.2: Average solution quality for instances with $\Lambda = 0\%$ depending on relative test instance size N.

	gap_1 in %		*gap_2* in %		*gap_3* in %	
N	MSA	S-VNS	MSA	S-VNS	MSA	S-VNS
100%	**7.34**	7.12	-0.86	-0.49	-0.19	1.07
200%	**8.25**	7.08	1.50	-1.45	1.50	0.40
300%	**10.79**	8.08	2.50	-1.79	2.36	-0.97
400%	**11.20**	9.29	2.04	-2.69	3.13	-1.64
avg.	**9.39**	7.89	1.30	-1.61	1.70	-0.28

Table 4.3: Average solution quality for instances with $\Lambda = 30\%$ depending on relative test instance size N.

results in this case. Both stochastic methods perform almost equally well on average. For smaller instances, dynamic S-VNS leads to larger improvements than the MSA, whereas the opposite is true for larger instances.

In the dynamic cases ($\Lambda > 0\%$), the solution quality obtained by both our stochastic solution approaches declines compared to the corresponding myopic approaches. This can be seen in Figure 4.8 for the MSA and Figure 4.9 for dynamic S-VNS. These figures, like Figures 4.4 and 4.5 show the average gap between the results obtained by the two compared methods (circles) and the corresponding 95% confidence intervals. The relative advantage of the MSA, however, decreases faster than the one of dynamic S-VNS. Regarding the slightly dynamic case ($\Lambda = 30\%$, Table 4.3), the MSA obtains solutions that are on average 9.39% better than the ones found by the MPA. Dynamic S-VNS can improve on the solutions found by dynamic VNS by 7.89%. In the medium dynamic case ($\Lambda = 50\%$, Table 4.4), the MSA achieves solutions that are on average 5.50% better than the ones obtained by the MPA whereas dynamic S-VNS yields 6.96% better results than dynamic VNS. In the highly dynamic case ($\Lambda = 80\%$, Table 4.5), dynamic S-VNS still obtains results that are on average 3.70% better than the ones found by dynamic VNS. The solutions found by the MSA are on average only 1.30% better than the ones found by the MPA.

Besides the direct comparison between our stochastic and myopic methods, a

	gap_1 in %		gap_2 in %		gap_3 in %	
N	MSA	S-VNS	MSA	S-VNS	MSA	S-VNS
100%	6.27	**6.94**	-0.42	-2.03	0.61	1.39
200%	3.30	**4.47**	-0.26	-2.94	0.51	-1.40
300%	5.46	**7.97**	0.52	-0.99	1.76	-0.64
400%	6.95	**8.46**	0.86	-2.53	1.39	-1.92
avg.	5.50	**6.96**	0.17	-2.12	1.07	-0.64

Table 4.4: Average solution quality for instances with $\Lambda = 50\%$ depending on relative test instance size N.

	gap_1 in %		gap_2 in %		gap_3 in %	
N	MSA	S-VNS	MSA	S-VNS	MSA	S-VNS
100%	2.39	**4.47**	-1.50	-2.42	-1.54	0.75
200%	1.83	**3.06**	-4.55	-0.77	-1.54	-0.57
300%	1.05	**3.63**	-1.49	-2.09	-0.75	-0.50
400%	-0.07	**3.63**	-1.99	-3.01	-0.86	-1.84
avg.	1.30	**3.70**	-2.38	-2.07	-1.17	-0.54

Table 4.5: Average solution quality for instances with $\Lambda = 80\%$ depending on relative test instance size N.

comparison between all four methods is given in Table 4.6. As can be seen, the MPA performs on average 2.47% and 2.22% worse than dynamic VNS in the static and slightly dynamic case, respectively. In the cases with more dynamic requests ($\Lambda >= 50\%$), both methods provide results of similar quality. The MSA can improve on the solutions found by dynamic VNS by 2.60% to 11.30% depending on the degree of dynamism, which is slightly worse than the improvements obtained by dynamic S-VNS. This indicates, that for the problem at hand, the concept of using a long term memory as in the MPA and the MSA seems to have no strong beneficial effect. However, as the direct comparison between the MPA and the MSA has shown, the potential benefits from using stochastic information in this concept are more sensitive to the degree of dynamism than in the case of dynamic S-VNS. In other words, although based on a conceptual design that is less sophisticated than the one used by the MPA and the MSA, the potential benefits from using stochastic information in the case of dynamic VNS and dynamic S-VNS appear to be more robust against changes in the degree of dynamism.

Another interesting aspect is the performance of our algorithms regarding the secondary and tertiary objective function. While dynamic S-VNS uses 1.96% more vehicles than dynamic VNS, the MSA can reduce the average number of vehicles used by 2.92% compared to the MPA in the static case ($\Lambda = 0\%$, Table 4.2). This

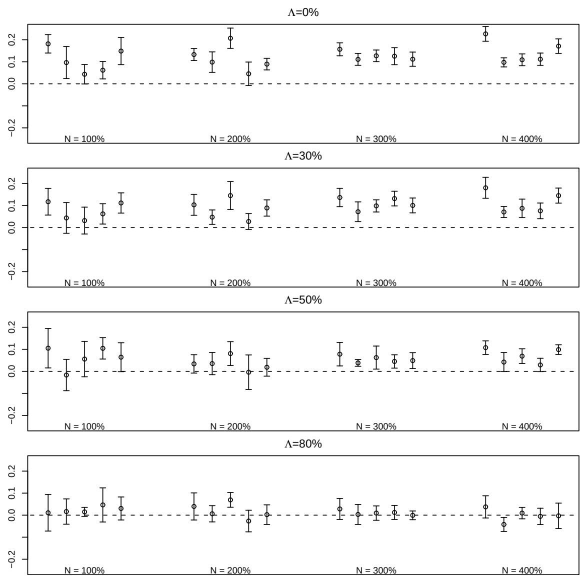

Figure 4.8: Average (circles) and 95% confidence intervals (whiskers) of the relative gaps in the primary objective obtained by the MPA with time-dependent travel speeds and the MSA (Λ - degree of dynamism, N - instance size).

difference, however, is caused by the fact that the MPA uses 7.98% more vehicles than dynamic VNS anyway. This effect is of similar magnitude for higher degrees of dynamism as well (see Tables 4.3,4.4, and 4.5). As can be seen from the same tables, both stochastic methods obtain solutions that have between -1.17% and 2.56% gap to the route durations found by the corresponding myopic approach. This means, that the benefits obtained by the stochastic methods can be achieved without large sacrifices regarding the secondary and tertiary objective values.

Finally, we can see from Tables 4.2, 4.3, 4.4 and 4.5, that the test instance size does also have an impact on relative solution quality. This influence, however, is relatively small except for the slightly dynamic situation ($\Lambda = 30\%$).

Our results show, that possible advantages of using stochastic information about future travel time deviations while planning is highly depending on two factors. On the one hand, the conceptual design of the used method does have a strong influence on the obtained results. On the other hand, the magnitude of possible improvements depends on the degree of dynamism. In detail, we found that our single solution based dynamic S-VNS approach is slightly less sensitive to the degree of dynamism than MSA. Still, both stochastic methods are in almost all cases able to benefit from the scarce information about the stochastic effects on travel speeds that is available.

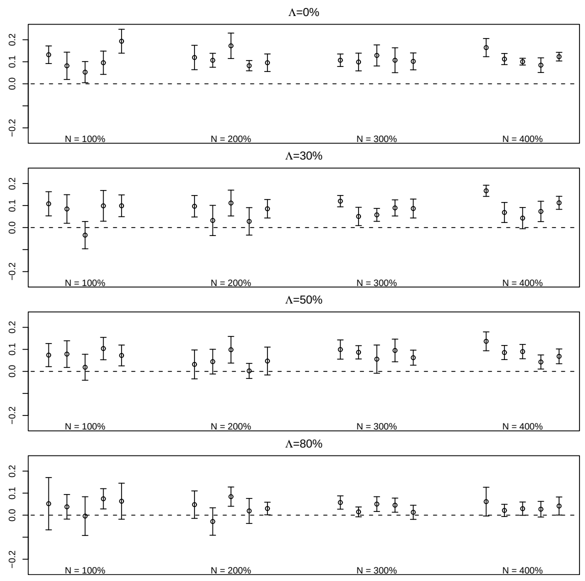

Figure 4.9: Average (circles) and 95% confidence intervals (whiskers) of the relative gaps in the primary objective obtained by VNS with time-dependent travel speeds and S-VNS (Λ - degree of dynamism, N - instance size).

4.5 Summary

In this chapter we adapt the two conceptually similar pairs of metaheuristic solution approaches for the DSDARP to the requirements of the dynamic dial-a-ride problem with stochastic time-dependent travel speeds (DDARPTD). The first pair consists of a dynamic variable neighborhood search method (dynamic VNS) and a stochastic variant thereof (dynamic S-VNS). The second pair includes the multiple plan approach (MPA) and the multiple scenario approach (MSA). All four methods use our implementation of dynamic VNS as a search component. Our main aim is to determine if the inclusion of information about future stochastic deviations from time-dependent travel speeds in stochastic algorithms leads to better solutions than using only average time-dependent travel speeds in the corresponding myopic methods.

We test all algorithms on sets of real world inspired test instances including a total of 20 instances. Our findings show, that in the static case ($\Lambda = 0\%$, i.e., all inbound requests known a priori) both stochastic approaches lead to remarkable average improvements of more than 10% over their myopic counterparts. In more dynamic situations ($\Lambda = \{30\%, 50\%, 80\%\}$), the relative performance of MSA deteriorates faster than the one of dynamic S-VNS. Still, both stochastic methods perform reasonably better than their myopic counterparts. Thus, the benefit from using information about stochastic deviations from future time-dependent travel

	gap_1 in %			gap_2 in %			gap_3 in %		
Λ	MPA	MSA	S-VNS	MPA	MSA	S-VNS	MPA	MSA	S-VNS
0%	-2.47	10.18	**11.30**	-7.98	-4.77	-1.96	-2.85	-0.19	-1.13
30%	-2.22	7.44	**7.89**	-2.50	-1.12	-1.61	-1.10	0.65	-0.28
50%	0.32	5.81	**6.96**	-1.64	-1.45	-2.12	-0.78	0.31	-0.64
80%	1.35	2.60	**3.70**	1.31	-0.99	-2.07	1.94	0.80	-0.54
avg.	-0.75	6.51	**7.46**	-2.70	-2.08	-1.94	-0.70	0.39	-0.65

Table 4.6: Average solution quality depending on degree of dynamism. All values relative to results obtained by dynamic VNS. (Λ - degree of dynamism)

speeds is potentially noteworthy, but it's magnitude highly depends on the problem characteristics (e.g., degree of dynamism).

In situations with low degrees of dynamism, MSA leads to larger improvements relative to the corresponding myopic approach than dynamic S-VNS. In cases with higher degrees of dynamism, the opposite is true. It seems that dynamic S-VNS is the more robust way to integrate stochastic information into the planning process. Additionally, the concept of using a long term memory in the solution process (as in MPA and MSA) seems not to be favorable over a pure VNS approach in the case of the dynamic DARP with stochastic time-dependent travel speeds.

5 Heterogenous aspects and multiple depots

5.1 Introduction and related work

In Chapter 3 we study the effects of exploiting stochastic information about future return transports while planning vehicle routes for the dynamic dial-a-ride problem. In Chapter 4 we do the same for stochastic time-dependent travel speeds. In this chapter, we combine these two stochastic aspects and include additional real world motivated constraints to the DDARP in order to obtain an even more realistic problem description. For this purpose, we assume stochastic information about both, future return transports and time-dependent travel speeds, to be available and additionally include heterogeneous patients, heterogeneous vehicles and multiple depots in the problem (see Section 5.2 for details). The resulting problem is denoted as heterogeneous dynamic stochastic dial-a-ride problem with stochastic time-dependent travel speeds (HDSDARPTD).

Among the first publications to study heterogeneous versions of the passenger transportation problem was a paper by Toth and Vigo [63]. The authors consider seated passengers, patients who require a wheelchair and different types of vehicles in the context of the handicapped persons transport problem which is a generalization of the pickup and delivery problem with time windows. The proposed solution methods include a parallel insertion heuristic and a tabu thresholding procedure used to improve the solutions found by the heuristic. Melachrinoudis et al. [64] present a heterogeneous extension to the DARP including vehicles with different capacities but only a single mode of transportation and solve it using a tabu search approach. A similar variant of the DARP including different vehicle capacities but a single mode of transportation is also studied by Rekiek et al. [65] using a grouping genetic algorithm. Beaudry et al. [66] present an adaptation of a tabu search approach for a heterogeneous version of the DARP for patient transportation operations inside hospitals. The studied problem is dynamic and includes different modes of transportation as well as different vehicle types. Another application in the context of dynamic heterogeneous intrahospital patient transportation is presented by Hanne et al. [67]. Kergosien et al. [68] study the dynamic problem when transporting patients between different care units of a hospital and propose a tabu search method with an adaptive memory. Finally, Parragh [69] presents exact and metaheuristic solution approaches for the DARP with four different modes of transportation and different types of vehicles. As this work is also considering the requirements of an

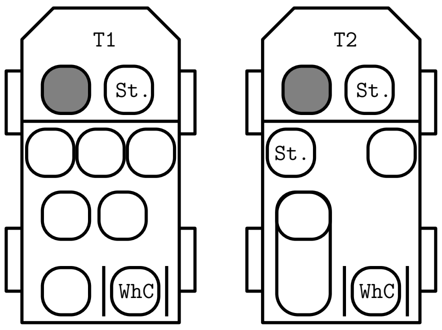

Figure 5.1: Vehicles of type 1 (left) and type 2 (right).

Passenger type	Staff seat	Patient seat	Stretcher	Wheelchair place
Accompanying person	X	X	X	
Seated patient		X	X	
Lying patient			X	
Patient with wheelchair				X

Table 5.1: Available upgrading options for the different transportation modes.

Austrian ambulance service provider, our study uses the same types of vehicles and patients.

5.2 Problem definition

The heterogeneous dynamic stochastic dial-a-ride problem with stochastic time-dependent travel speeds is a combination and extension of the dynamic stochastic dial-a-ride problem with expected return transports and the dynamic dial-a-ride problem with stochastic time-dependent travel speeds. This means, that we assume stochastic information about both, future return transports and time-dependent travel speeds, to be available and additionally include heterogeneous patients, heterogeneous vehicles and multiple depots in the problem. To be precise, we no longer assume every patient and every vehicle to be the same. Instead, we assume that there are four different types of resources depending on the required mode of transportation: staff seats, patient seats, stretchers and wheelchair places. The presented types of vehicles and patients are taken from Parragh [69]. In addition to this, we also no longer assume all vehicles to be located at a common home depot at the beginning of the planning horizon. Instead, we assume each vehicle do be located at a specific hospital site to which it must return in the evening.

Some patients require a normal passenger seat as they can be transported sitting (due to legal restrictions using a staff seat is not allowed for patients in Austria).

Alternatively, such patients can, however, be transported lying on a stretcher (which is denoted as upgrading). Some patients need to be transported in a lying position which means that they require a stretcher to be available in the vehicle used to service their request. A third type of patients requires a wheelchair place to be available in order to be transported. The latter two types of patients cannot be upgraded. Finally, every patient may be accompanied by an additional person. Such an accompanying person normally uses a staff seat, but also a patient seat or a stretcher can be used if no free staff seat is available. Additionally, we assume that two different types of vehicles are available for transporting the patients. Vehicles of type 1 are equipped with one staff seat, six patient seats, no stretcher and one wheelchair place. Vehicles of type 2 are equipped with two staff seats, one patient seat, one stretcher and one wheelchair place. A graphical representation of these two vehicle types is shown in Figure 5.1 taken from Parragh [69]. The possible options for upgrading the different transportation modes are shown in Table 5.1 which is also taken from Parragh [69].

The remainder of the HDSDARPTD is a combination of the DSDARP and the DDARPTD. This means, that we assume each outbound request r to have a specific probability R_r to cause an inbound request during the planning period (as in the case of the DSDARP). For outbound requests we still assume that no a priori information is known. Additionally, the problem is defined on a (directed) real world road network with stochastic time-dependent travel speeds as is the DDARPTD. Therefore, we assume that going from node A to any other node B takes $\hat{T}(t, A, B) = \hat{T}_{avg}(t, A, B) + T_{stoc}(t, A, B)$ time units. $\hat{T}_{avg}(t, A, B)$ is the time required to travel from A to B when leaving A at a given time t and is based on the average vehicle speeds during the affected time intervals which are assumed to be known a priori. Again, the term $T_{stoc}(t, A, B)$ represents the stochastic influence on this travel time which is revealed only upon departure from A.

As the travel speeds used for the HDSDARPTD are time-dependent, the maximum detour constraint introduced for the DDARPTD is used for this problem as well. We define the time required to go directly from p_r to d_r departing at time t as $\hat{T}_{direct}(t, p_r, d_r)$. The time between the planned end of service at p_r and the planned start of service at d_r is denoted as T_{real}. Then, the equation $T_{real} \leq \hat{T}_{direct}(t, p_r, d_r) + 30$ should not be violated. Due to the stochastic influence on travel speeds, we again cannot guarantee that this equation will strictly hold and penalize this ride time violation in the objective function. Note, that waiting times may only be planned when a vehicle is empty.

For the HDSDARPTD we use the same lexicographic objective function as for the DDARPTD as the aim of planning the vehicle routes is still to minimize passenger dissatisfaction while keeping total costs as low as possible. The primary objective is to minimize the sum of tardiness, earliness and ride time violations over all routes. The secondary objective is the number of routes (vehicles used). The third objective is the total route duration. In general, solutions are compared according to the primary objective. In case two solutions are equal in terms of the primary objective, the secondary one is used for comparison. Only if both, primary and secondary objective, are equal for two solutions, comparison is based on the third objective.

5.3 Solution methods

The aim of the research presented in this chapter is twofold. First, we want to find out if the combination of the two stochastic aspects studied separately in Chapters 3 and 4 leads to effects different from the ones found before. Second, we want to study the effect of additionally introducing heterogenous aspects into the problem the Austrian ambulance service providers are facing. For this purpose, we use the same solution approaches presented for the DSDARP and the DDARPTD to solve the HDSDARPTD.

Again, we need to adapt our four solution methods to the requirements of this extended problem. This means, that we need to combine the modifications made to solve the DSDARP and the ones used for the DDARPTD in order to obtain solution methods capable of handling stochastic return transports as well as stochastic time-dependent travel speeds. Additionally, we need to adapt all methods to the heterogeneous aspects of the new problem. To be precise, the methods need to make sure, that vehicles used to service a specific request really possess the capacity required by this request with respect to each of the four transportation modes.

5.3.1 Simulation framework

As we combine the two extensions to the DDARP presented in the previous chapters, we also need to adapt the simulation framework accordingly. This means, that we merge the features of both simulators into one while extending the framework in order to support multiple resources and different vehicle types. The resulting framework still loads and manages all problem specific information like, for example, test instance data, distance matrices, vehicle information, and request arrival lists. It also has the interface which allows the solver modules to request expected travel times between two locations for a given departure or arrival time. In this way the framework provides travel times based on the average speeds within the affected intervals. Again, we assume that travel times including the actual stochastic influences are revealed to the solver modules only upon departure of the respective vehicle. The framework also takes care of the FIFO property as we use the same block scheduling algorithm as presented in Section 4.3.2. Further modifications are required in order to handle the HDSDARPTD because of the heterogeneous vehicle fleet, the heterogeneous patients and the use of multiple depots.

5.3.2 Scheduling algorithm

As the heterogeneous dynamic stochastic dial-a-ride problem with stochastic time-dependent travel speeds includes multiple resources and different patient types, we

Algorithm 17 Heterogeneous block scheduling algorithm: Phase 1

```
1: $i \leftarrow 0$
2: $Q_0 \leftarrow 0$
3: for $n = \{1, \dots, |S|\}$ do
4:     $A_n \leftarrow D_{n-1} + \hat{T}(D_{n-1}, n-1, n)$
5:     $B_n \leftarrow A_n$
6:     $D_n \leftarrow B_n + d_n$
7:     if CheckCapacities(n) then
8:         return Infeasible
9:     end if
10:    if VehicleEmptyBeforeStop($n$) $\wedge (A_n < e_n)$ then
11:        $A_n \leftarrow e_n$
12:        $D_{n-1} \leftarrow A_n - \check{T}(A_n, n-1, n)$
13:        $B_n \leftarrow A_n$
14:        $D_n \leftarrow B_n + d_n$
15:        $i \leftarrow i + 1$
16:        $\Theta_i^{start} \leftarrow n$
17:        $\Theta_i^{waiting} \leftarrow D_{n-1} - B_{n-1} - d_{n-1}$
18:        $\Theta_i^{earliness} \leftarrow \max\{e_n - B_n, 0\}$
19:        $\Theta_i^{slack} \leftarrow \max\{l_n - B_n, 0\}$
20:    else
21:        $\Theta_i^{earliness} \leftarrow \max\{\max\{e_n - B_n, 0\}, \Theta_i^{earliness}\}$
22:        $\Theta_i^{slack} \leftarrow \min\{\max\{l_n - B_n, 0\}, \Theta_i^{slack}\}$
23:    end if
24: end for
```

need to adapt our block scheduling algorithm accordingly. Given a specific sequence of transportation requests inside a route, the feasibility and timing of the route is determined. In order to correctly handle the heterogeneous aspects of the problem, the algorithm needs to match the available capacities with the requirements of each transported patient.

The outline of phase 1 of the block scheduling algorithm capable of handling the heterogeneous aspects of the HDSDARPTD is presented in Algorithm 17. As the sequence of the stops is not altered during the scheduling process, phase 2 of the BSA remains unaffected by the heterogeneous aspects of the problem (see Algorithm 12). In order to make this outline easier to read, we condense all the capacity checks into a single function call (Line 7).

For this purpose, we extend the notation used in Section 4.3.2 as follows. S is again the set of nodes to be visited by the current route (in chronological order, 0 and $|S|$ are the indices of the depot nodes). Q_0 is the number of staff seats available inside the vehicle servicing the current route. Q_1, Q_2 and Q_3 is the number of installed patient seats, the number of stretchers and the number of wheelchair

Algorithm 18 Heterogeneous block scheduling algorithm: capacity check pickup

```
1: if $\mathcal{Q}_0 + q_{0n} <= Q_0$ then
2:     $\mathcal{Q}_0 \leftarrow \mathcal{Q}_0 + q_{0n}$
3: else if $\mathcal{Q}_1 + q_{0n} <= Q_1$ then
4:     $\mathcal{Q}_{01} \leftarrow \mathcal{Q}_{01} + q_{0n}$
5:     $\mathcal{Q}_1 \leftarrow \mathcal{Q}_1 + q_{0n}$
6: else if $\mathcal{Q}_2 + q_{0n} <= Q_2$ then
7:     $\mathcal{Q}_{02} \leftarrow \mathcal{Q}_{02} + q_{0n}$
8:     $\mathcal{Q}_2 \leftarrow \mathcal{Q}_2 + q_{0n}$
9: else
10:     return Infeasible
11: end if
12: if $\mathcal{Q}_1 + q_{1n} <= Q_1$ then
13:     $\mathcal{Q}_1 \leftarrow \mathcal{Q}_1 + q_{1n}$
14: else if $\mathcal{Q}_2 + q_{1n} <= Q_2$ then
15:     $\mathcal{Q}_{12} \leftarrow \mathcal{Q}_{12} + q_{1n}$
16:     $\mathcal{Q}_2 \leftarrow \mathcal{Q}_2 + q_{1n}$
17: else
18:     return Infeasible
19: end if
20: if $\mathcal{Q}_2 + q_{2n} <= Q_2$ then
21:     $\mathcal{Q}_2 \leftarrow \mathcal{Q}_2 + q_{2n}$
22: else
23:     return Infeasible
24: end if
25: if $\mathcal{Q}_3 + q_{3n} <= Q_3$ then
26:     $\mathcal{Q}_3 \leftarrow \mathcal{Q}_3 + q_{3n}$
27: else
28:     return Infeasible
29: end if
```

places available, respectively. The values $\mathcal{Q}_0, \mathcal{Q}_1, \mathcal{Q}_2$ and $\mathcal{Q}_3$ represent the used amount of the respective resource when arriving at the current stop. In order to handle the upgrading options correctly, we additionally introduce $\mathcal{Q}_{01}$ and $\mathcal{Q}_{02}$ as the number of accompanying persons upgraded to a patient seat or a stretcher. Also, $\mathcal{Q}_{12}$ is the number of sitting patients transported on a stretcher instead of using a patient seat. Note, that lying patients and wheelchair users cannot be upgraded. The amount of resources required by each of the stops along the route are denoted as q_{0n}, q_{1n}, q_{2n} and q_{3n}, respectively (assuming the pickup and delivery stop of the same patient to have the same non-negative capacity requirements).

The outline of this function for pickup stops is presented in Algorithm 18. First, the function checks if the vehicle can accommodate an additional accompanying

Algorithm 19 Heterogeneous block scheduling algorithm: capacity check delivery

```
1: if $\mathcal{Q}_{02} > 0$ then
2:     $\mathcal{Q}_{02} \leftarrow \mathcal{Q}_{02} - q_{0n}$
3:     $\mathcal{Q}_2 \leftarrow \mathcal{Q}_2 - q_{0n}$
4: else if $\mathcal{Q}_{01} > 0$ then
5:     $\mathcal{Q}_{01} \leftarrow \mathcal{Q}_{01} - q_{0n}$
6:     $\mathcal{Q}_1 \leftarrow \mathcal{Q}_1 - q_{0n}$
7: else
8:     $\mathcal{Q}_0 \leftarrow \mathcal{Q}_0 - q_{0n}$
9: end if
10: if $\mathcal{Q}_{12} > 0$ then
11:     $\mathcal{Q}_{12} \leftarrow \mathcal{Q}_{12} - q_{1n}$
12:     $\mathcal{Q}_2 \leftarrow \mathcal{Q}_2 - q_{1n}$
13: else
14:     $\mathcal{Q}_1 \leftarrow \mathcal{Q}_1 - q_{1n}$
15: end if
16: $\mathcal{Q}_2 \leftarrow \mathcal{Q}_2 - q_{2n}$
17: $\mathcal{Q}_3 \leftarrow \mathcal{Q}_3 - q_{3n}$
```

person if required (Lines 1 to 11). Second, it checks the available space for sitting patients (Lines 12 to 19) followed by the space for lying patients (Lines 20 to 24) and wheelchair users (Lines 25 to 29). For delivery stops, the function needs to make sure that all resources are freed correctly (Algorithm 19). This means, that upgraded persons need to be downgraded as soon as possible in order to free the more scarce resources (i.e., stretchers, patient seats).

5.3.3 Dynamic variable neighborhood search

We adapt the dynamic variable neighborhood search algorithm presented in the previous chapters in order to meet the requirements of the HDSDARPTD. We use the adaptation of our dynamic VNS presented for the DDARPTD (see Section 4.3.3) as a basis for our modifications as it already includes all changes required to handle time-dependent travel speeds. Due to the heterogeneous aspects of the problem the algorithm needs to be slightly modified in order to guarantee the matching of transportation modes (sitting, lying, in wheelchair and accompanying persons) to the available resources of the vehicles (patient seats, stretchers, wheelchair places and staff seats). In the outline given in Algorithm 20, however, these modifications cannot be seen as they do not affect this rather abstract outline of the algorithm but more fundamental parts of the implementation. To be precise, especially the scheduling algorithm needs to be modified in order to correctly handle multiple resources and heterogeneous patients. For this purpose, we include the respective modifications in the block scheduling algorithm as described in Section 5.3.2.

The dynamic VNS algorithm starts by creating an initial feasible solution includ-

Algorithm 20 Structure of dynamic VNS for the HDSDARPTD

```
1: $x \leftarrow$ InitialSolution()
2: $\mathcal{N}(\kappa) \leftarrow$ SelectFirstNeighborhood()
3: while StoppingCriterionNotMet() do
4:     $x \leftarrow$ RescheduleWithTrueTravelSpeeds($x$)
5:     $x \leftarrow$ InsertNewRequests($x$)
6:     $x' \leftarrow$ ShakeSolution($x$,$\mathcal{N}(\kappa)$)
7:     $x \leftarrow$ MoveOrNot($x, x'$)
8:     $\mathcal{N}(\kappa) \leftarrow$ SelectNextNeighborhood($\kappa$)
9: end while
10: return $x$ as best found solution
```

ing all static request using the modified cheapest insertion heuristic described in Section 3.3.2 and then enters the iterative search process. At the beginning of each iteration, the current incumbent solution needs to be updated with respect to the current traffic situation (Line 4). Note, that this is only required if a vehicle departed since the last update (as we assume true travel speeds to be revealed upon departure of a vehicle). Then, all new transportation requests are inserted into the current incumbent solution using the cheapest insertion method (Line 5). The current incumbent solution is then perturbed by a shaking step using the neighborhood operators described in Section 3.3.3. Finally, the algorithm decides whether or not to accept the new solution as current incumbent solution and selects the next neighborhood operator and intensity level accordingly (Lines 7 and 8).

5.3.4 Dynamic stochastic variable neighborhood search

In order to adapt the dynamic S-VNS to the new requirements of the HDSDARPTD we combine the two variants of dynamic S-VNS presented for the DSDARP and the DDARPTD. This means, that we use the general outline of the version tailored to the DDARPTD as a basis and introduce the required changes in order to also exploit the stochastic information about future return transports. All modifications required to handle the heterogeneous extensions are already included in the block scheduling algorithm.

An outline of the modified dynamic S-VNS method is given in Algorithm 21. In the case of the HDSDARPTD, the current incumbent solution is evaluated based on a sample of future travel speeds and a sample of future return transports at the same time. This means that the algorithm determines possible future deviations from each road segment's average travel speed for each future interval by sampling the corresponding distribution parameters (Line 3) and re-schedules the initial solution using this information (Line 4). At the beginning of each iteration the algorithm adapts the current incumbent solution to stochastic travel speeds if they are known by now (Line 8). It then inserts all new requests which have become known during the last iteration into this solution (Line 9). This is followed by a shaking step using

Algorithm 21 Structure of dynamic S-VNS for the HDSDARPTD

```
1: $x \leftarrow$ InitialSolution()
2: $\mathcal{N}(\kappa) \leftarrow$ SelectFirstNeighborhood()
3: $Z_{speeds} \leftarrow$ SampleFutureTravelSpeeds()
4: $x \leftarrow$ Reschedule($x, Z_{speeds}$)
5: $u, i \leftarrow 0$
6: **while** StoppingCriterionNotMet() **do**
7:    $i \leftarrow i + 1$
8:    $x \leftarrow$ RescheduleWithTrueTravelSpeeds($x$)
9:    $x \leftarrow$ InsertNewRequests($x$)
10:   $x' \leftarrow$ ShakeSolution($x, \mathcal{N}(\kappa)$)
11:   $Z_{requests} \leftarrow$ SampleFutureRequests($1, S_{max}$)
12:   $x \leftarrow$ InsertSampledRequests($x, Z_{requests}$)
13:   $x' \leftarrow$ InsertSampledRequests($x', Z_{requests}$)
14:   **if** $i - u > U$ **then**
15:      $u \leftarrow i$
16:      $Z_{speeds} \leftarrow$ SampleFutureTravelSpeeds()
17:      $x \leftarrow$ Reschedule($x, Z_{speeds}$)
18:      $x' \leftarrow$ Reschedule($x', Z_{speeds}$)
19:   **end if**
20:   $x \leftarrow$ MoveOrNot($x, x'$)
21:   $x \leftarrow$ RemoveSampledRequestsFromSolution($x, Z_{requests}$)
22:   $\mathcal{N}(\kappa) \leftarrow$ SelectNextNeighborhood($\kappa$)
23: **end while**
24: **return** $x$ as best found solution
```

the same neighborhood structures as already described for dynamic VNS (Line 10). Following the shaking step, a sample of future return transports is generated and inserted into the current incumbent and the candidate solution (Lines 11 to 13). Then, it re-schedules both, the current incumbent and the candidate solution, accordingly (Lines 14 to 19). After every U iterations, the algorithm updates the used sample of future travel speeds. The solution which has the better sample average estimator with respect to the current samples is selected as new current incumbent solution after removing the sampled return transports from it again (Line 20). We use a single sample of future travel speeds and a single sample of future return transports for this evaluation. It is again essential to keep the sampled information about future travel speeds included in the solution that is to be executed in order to obtain significant benefits over an algorithm that uses only average travel speeds for planning.

5.3.5 Multiple plan approach

Our second pair of metaheuristics we adapt to the requirements of the HDSDARPTD

Algorithm 22 Structure of the MPA for the HDSDARPTD

```
1: $P \leftarrow$ InitialSolution()
2: $\mathcal{N}(\kappa) \leftarrow$ SelectFirstNeighborhood()
3: while StoppingCriterionNotMet() do
4:     $\bar{x} \leftarrow$ RescheduleWithTrueTravelSpeeds($\bar{x}$) $\forall \bar{x} \in P$
5:     $x \leftarrow$ SelectCurrentIncumbent($P$)
6:     for $\bar{x} \in P$ do
7:         if $(\bar{x} \neq x) \wedge$ (Timeout($\bar{x}, x$) $\vee$ Departure($\bar{x}, x$)) then
8:             $P \leftarrow P \setminus \{\bar{x}\}$
9:         else
10:            InsertNewRequests($\bar{x}$)
11:        end if
12:    end for
13:    $x' \leftarrow$ ShakeSolution($x, \mathcal{N}(\kappa)$)
14:    if $x' \notin P$ then
15:        $P \leftarrow P \cup \{x'\}$
16:    end if
17:    $\mathcal{N}(\kappa) \leftarrow$ SelectNextNeighborhood($\kappa$)
18: end while
```

is based on the multiple plan approach. As the variant of the MPA presented for the DDARPTD is already able to handle time-dependent travel speeds and dynamic requests (return transports are treated as being dynamic as this is a myopic solution approach), we do not need to include additional modifications with respect to these two aspects. The presence of heterogeneous aspects, however, needs to be taken into account by the method. The changes required in order to handle the different vehicle and patient types are already included in the scheduling algorithm and thus do not show up in the general outline of the MPA given in Algorithm 22.

The main idea behind the MPA still is to use a pool of solutions as long term memory in which each unique solution found during the search process is stored. At every point in time, the algorithm uses one of these solutions as a current incumbent solution which is to be executed by the vehicles (Line 5). To guarantee the feasibility of all solutions in the pool, solutions which are incompatible with decisions made in the current incumbent solution are eliminated from the pool whenever necessary (Line 8). This way, the resulting solution is defined by the sequence of actions taken during execution. As the HDSDARPTD is structurally similar to the DSDARP and the DDARPTD, we decide to use the best solution in the pool as our current incumbent solution as we do for these two problems.

5.3.6 Multiple scenario approach

Finally, we need to merge the two variants of the multiple scenario approach presented for the DSDARP and the DDARPTD into one solution method for the

Algorithm 23 Structure of the MSA for the HDSDARPTD

```
 1: $P \leftarrow$ InitialSolution()
 2: $\mathcal{N}(\kappa) \leftarrow$ SelectFirstNeighborhood()
 3: $Z_{speeds} \leftarrow$ SampleFutureTravelSpeeds()
 4: $\bar{x} \leftarrow$ Reschedule($\bar{x}, Z_{speeds}$) $\forall \bar{x} \in P$
 5: $u, i \leftarrow 0$
 6: **while** StoppingCriterionNotMet() **do**
 7:    $i \leftarrow i + 1$
 8:    **if** $i - u > U$ **then**
 9:       $u \leftarrow i$
10:       $Z \leftarrow$ SampleFutureTravelSpeeds()
11:       $\bar{x} \leftarrow$ Reschedule($\bar{x}, Z_{speeds}$) $\forall \bar{x} \in P$
12:    **end if**
13:    $\bar{x} \leftarrow$ RescheduleWithTrueTravelSpeeds($\bar{x}$) $\forall \bar{x} \in P$
14:    $x \leftarrow$ SelectCurrentIncumbent($P$)
15:    **for** $\bar{x} \in P$ **do**
16:       **if** $(\bar{x} \neq x) \wedge$ (Timeout($\bar{x}, x$) $\vee$ Departure($\bar{x}, x$)) **then**
17:          $P \leftarrow P \setminus \{\bar{x}\}$
18:       **else**
19:          InsertNewRequests($\bar{x}$)
20:       **end if**
21:    **end for**
22:    $Z_{requests} \leftarrow$ SampleFutureRequests($1, S_{max}$)
23:    $x' \leftarrow$ AddSampledRequestsToSolution($x, Z_{requests}$)
24:    $x' \leftarrow$ ShakeSolution($x', \mathcal{N}(\kappa)$)
25:    $x' \leftarrow$ RemoveSampledRequestsFromSolution($x', Z_{requests}$)
26:    **if** $x' \notin P$ **then**
27:       $P \leftarrow P \cup \{x'\}$
28:    **end if**
29:    $\mathcal{N}(\kappa) \leftarrow$ SelectNextNeighborhood($\kappa$)
30: **end while**
```

HDSDARPTD. This means, that we use the variant described for the DDARPTD as a basis for our modifications and include the parts required in order to handle stochastic return transports. An outline of this modification is shown in Algorithm 23.

The algorithm starts by creating an initial solution using the modified cheapest insertion method presented in Section 3.3.2 and re-schedules this solution using a first sample of future travel speeds. This is followed by an iterative search phase. Every U iterations, all stored solutions are re-scheduled using a new sample of future travel speeds (Lines 8 to 12). Then, all solutions are re-scheduled if new true travel speeds have been revealed (i.e., if a vehicle has departed since the last iteration;

Line 13). Note, that the information of the current sample of future travel speeds is not removed from the solutions by doing so (if true travel speeds are still unknown, the sampled ones are used for re-scheduling). The current incumbent solution is selected out of the set of solutions (Line 14). After that, all solutions which are no longer compatible with the current incumbent solution are removed from the set (Line 17). New requests are inserted into all remaining solutions in the set using the modified cheapest insertion method (Line 19). Future requests are sampled and inserted into the current incumbent solution before the shaking step and removed again after the shaking was performed (Lines 23 to Line 25). This is done in order to obtain a solution including gaps in the schedule of the request, which can at a later point in time be used to accommodate real future requests more easily. The resulting solution is then stored in the set of solutions if it was not known before (Line 27) and the next neighborhood operator is selected accordingly (Line 29). Hereby, the same neighborhood structures are used as with all our other methods.

5.4 Computational experiments

The aim of studying the HDSDARPTD is to find out if exploiting stochastic information about two different problem aspects simultaneously in a heterogeneous setting leads to new insights. For this purpose, we adapt the test instances created for the DDARPTD (see Section 4.4.1) by re-including the stochastic return transports and adding heterogenous vehicles, heterogeneous patients and multiple depots. All computational experiments are performed on one core of a SUN Fire X2270 server with 2 quad-core Intel Xeon X5550 processors (2.66 GHz) with 24 GB of shared memory. The algorithms are implemented using C++ and the GNU compiler g++ in its version 4.1.2 on CentOS 5.5.

5.4.1 Test instances

Based on the test instances created for the DDARPTD (see Section 4.4.1), we create an extended set of instances for the HDSDARPTD. The main difference is, that we include the heterogeneous aspects of the problem into the instances. Additionally, we re-include the stochastic return transports from the DSDARP.

For the problem at hand we use only 5 of the largest test instances (defined by $N = 400\%$) and add the heterogeneous aspects as follows. First, we adapt the vehicle fleet defined for each instance. To be precise, we allow using up to 120 vehicles for solving each instance and assign a random number in the interval $[0, 1]$ to each of them. This number can then be used to define the vehicle type when loading the instance. Note, that we perform all our computational experiments using 50% chance for each vehicle type (i.e., an approximately equal number of vehicle of both types). Additionally, each vehicle is assigned a specific depot location by randomly selecting one of the ten largest hospital locations (with respect to the number of transportation requests going there according to the available real world data).

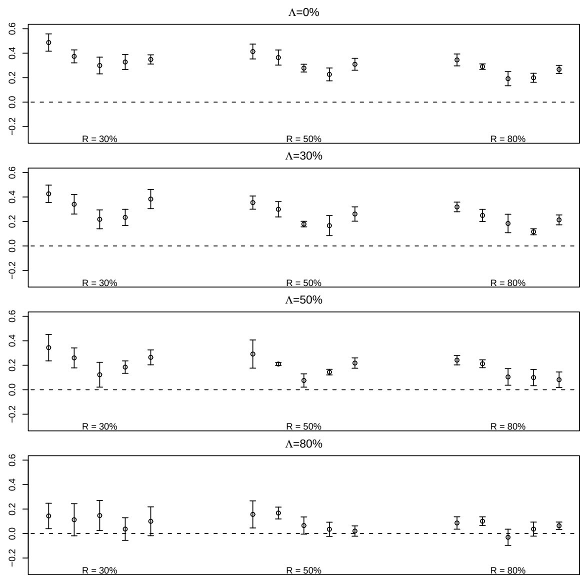

Figure 5.2: Average (circles) and 95% confidence intervals (whiskers) of the relative gaps in the primary objective obtained by the MPA with time-dependent travel speeds and the MSA (Λ - degree of dynamism, R - return transport probability).

Furthermore, we assign each transportation request a random number in the interval [0, 1] allowing to specify the capacity requirements of the respective patient when loading the instance data. For all our computational experiments we use the same parameters for the heterogeneous patients as presented by Parragh [69]. Namely, we set 50% of all patients to be accompanied by an additional person, 83% of the patients to be transported sitting, 11% to be transported lying and 6% to be using a wheelchair. These parameters as well as the used fleet size roughly correspond to the real situation faced by one of the largest Austrian ambulance service providers.

In our computational experiments, each of the test instances is solved using different settings for the return transport probability R and the amount of dynamic requests Λ (the proportion of inbound requests that are dynamic). The return transport probability can be $R = \{30\%, 50\%, 80\%\}$, whereas the degree of dynamism can be $\Lambda = \{0\%, 30\%, 50\%, 80\%\}$. Thus, we test 12 different parameter combinations for each of the 5 instances.

5.4.2 Results

In what follows, we present the results obtained during 5 independent runs on our test instances in terms of relative solution quality. This means, that results for the stochastic methods are given as percentage relative to the results obtained by

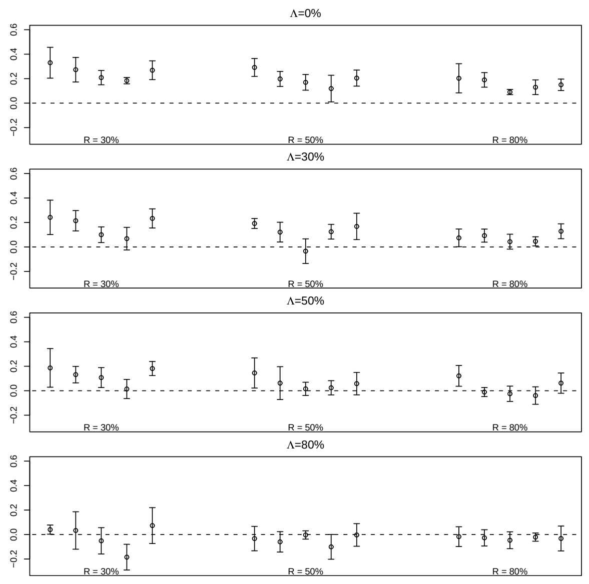

Figure 5.3: Average (circles) and 95% confidence intervals (whiskers) of the relative gaps in the primary objective obtained by VNS with time-dependent travel speeds and S-VNS (Λ - degree of dynamism, R - return transport probability).

their myopic counterparts. A positive percentage indicates that the corresponding method yields a better solution quality than the approach it is compared to, whereas a negative value indicates the opposite. All results are obtained using an update frequency of $U = 100$ iterations for the samples of our stochastic methods.

The main focus of our research is to find out if our stochastic algorithms (dynamic S-VNS and the MSA) are able to obtain better solutions for this problem than our myopic methods (dynamic VNS and the MPA). Especially the question if the observed effects differ from the ones found during the separate studies of the two stochastic effects is of relevance to us. A summary of all results obtained for $R = \{30\%, 50\%, 80\%\}$ depending on the degree of dynamism Λ is shown in Tables 5.2, 5.3, and 5.4, respectively. The columns titled gap_1 relate to the primary objective (sum of tardiness, earliness and ride time violations), the columns titled gap_2 and gap_3 relate to the secondary (number of vehicles used) and tertiary (total route duration) objective functions, respectively. Note, that we use a lexicographic objective function. This means that only the primary objective function can be used directly to compare solution quality. Results for the secondary and tertiary objective function are presented for the sake of completeness only. Figure 5.2 shows the average gap (and the 95% confidence intervals) between the solution quality obtained by the MSA and the one obtained by the MPA for each of the test instances. Figure 5.3 shows the same information for dynamic S-VNS and dynamic VNS.

	gap_1 in %		gap_2 in %		gap_3 in %	
Λ	S-VNS	MSA	S-VNS	MSA	S-VNS	MSA
0%	25.30	**36.75**	0.00	2.34	-4.27	4.29
30%	17.14	**32.00**	-5.61	1.27	-5.71	2.65
50%	12.43	**23.53**	-6.52	1.40	-7.18	0.52
80%	-1.79	**10.80**	-9.83	-2.97	-10.91	-1.22
avg.	13.27	**25.77**	-5.49	0.51	-7.02	1.56

Table 5.2: Average solution quality obtained for a return transport probability of $R = 30\%$ depending on degree of dynamism Λ.

	gap_1 in %		gap_2 in %		gap_3 in %	
Λ	S-VNS	MSA	S-VNS	MSA	S-VNS	MSA
0%	19.67	**31.84**	-1.29	2.02	-4.40	3.47
30%	11.41	**25.19**	-5.24	3.99	-5.87	3.70
50%	6.11	**18.79**	-6.24	1.02	-6.64	1.38
80%	-4.01	**8.89**	-9.70	-1.00	-9.21	-0.24
avg.	8.29	**21.18**	-5.62	1.51	-6.53	2.08

Table 5.3: Average solution quality obtained for a return transport probability of $R = 50\%$ depending on degree of dynamism Λ.

In the case with a low return transport probability ($R = 30\%$, Table 5.2), the average improvement obtained by the MSA is 25.77%, while the one achieved by dynamic S-VNS is 13.27%. This means, that both methods yield remarkable improvements over their myopic counterparts, but the MSA performs clearly better than dynamic S-VNS. In the case with 50% return transport probability (Table 5.3), the average improvement obtained by the MSA is still 21.18%, while the one obtained by dynamic S-VNS drops to 8.29%. In the case with 80% return transport probability, the situation is even worse. While the MSA can still achieve an average improvement of 16.85% over the MPA, dynamic S-VNS performs only 5.56% better than dynamic VNS. In addition to that, the MSA is able to obtain an improvement in all tested cases, while dynamic S-VNS performs even worse than dynamic VNS if the degree of dynamism is set to 80%.

The first observation is, that both, the return transport probability R and the degree of dynamism Λ, have a strong effect on the benefits achievable by exploiting the available stochastic information. While increasing the return transport probability causes a relatively small decrease in the obtained relative solution quality, the impact of increasing the degree of dynamism is remarkably larger. Both effects are basically caused by the same mechanism. Increasing the return transport probability R causes more return transports to occur, thus increasing the total de-

	gap_1 in %		gap_2 in %		gap_3 in %	
Λ	S-VNS	MSA	S-VNS	MSA	S-VNS	MSA
0%	15.28	**25.87**	-1.31	2.51	-3.25	3.42
30%	7.67	**21.61**	-6.03	2.64	-4.77	3.37
50%	2.18	**14.80**	-6.74	0.83	-6.18	1.77
80%	-2.88	**5.13**	-9.60	-1.36	-8.25	-0.47
avg.	5.56	**16.85**	-5.92	1.16	-5.61	2.02

Table 5.4: Average solution quality obtained for a return transport probability of $R = 80\%$ depending on degree of dynamism Λ.

gree of dynamism of the problem (part of which is compensated by exploiting the available stochastic information about these requests). Increasing Λ directly leads to more completely dynamic requests, thus giving the myopic approaches a relative advantage over their stochastic counterparts.

The second observation is, that using stochastic information about both stochastic influence factors simultaneously seems to lead to more stable benefits than exploiting each of the factors on its own (compare the results presented in Sections 3.4.4 and 4.4.2). A possible reason for this could be synergy effects. This means, that while one stochastic aspect has a strong beneficial effect for a specific test instance, the other might be less helpful. For another instance, it might be the other way round. Therefore, exploiting both aspects at the same time allows the algorithm to benefit in both cases.

Also, the magnitude of improvement achievable by exploiting both stochastic aspects simultaneously is much larger than the one obtainable by exploiting each of the two aspects separately (again, compare the results presented in Sections 3.4.4 and 4.4.2). However, the magnitude of the effect appears to be less than the sum of both effects. This might be due to the heterogeneous aspects included in the HDSDARPTD.

Our results show, that possible advantages of exploiting the stochastic information about future travel speed deviations and future return transports simultaneously while planning is highly depending on the total degree of dynamism. On the one hand, the actual amount of dynamic inbound requests has a strong influence on the resulting relative solution quality. On the other hand, the probability for return transports causes an effect in the same direction (which is however not as strong as the first one). Additionally, the conceptual design of the solution method is again very important. In the case of the HDSDARPTD, the MSA is clearly the more favorable approach.

5.5 Summary

In this chapter we present adapted versions of our four solution approaches which are capable of exploiting both stochastic aspects presented in the previous two chapters simultaneously. Additionally, the methods are able to handle heterogeneous patient requests, a heterogeneous vehicle fleet as well as multiple depots included in the heterogeneous dynamic stochastic dial-a-ride problem with stochastic time-dependent travel speeds (HDSDARPTD). We extend the dynamic variable neighborhood search (dynamic VNS), the dynamic stochastic VNS (S-VNS), the multiple plan approach (MPA) and the multiple scenario approach (MSA) to the requirements of this new problem class.

We test all four algorithms on a set of real world inspired test instances tailored for the HDSDARPTD. The performed computational experiments show that the multiple scenario approach clearly outperforms the dynamic S-VNS with respect to the obtained solution quality relative to the corresponding myopic approach. The average improvements obtained by the MSA are between 25.77% and 16.85% depending on the probability of return transports to occur. Additionally, the improvements obtained for the HDSDARPTD prove to be much more stable than the ones obtained for the DSDARP or the DDARPTD, indicating that increasing the amount of stochastic information that can be exploited has a positive effect on the robustness of the obtained results.

Furthermore, the total degree of dynamism present in a problem setting (depending primarily on the number of dynamic requests, but also on the probability for return transports) has a very strong influence on the benefits that can be obtained by exploiting the available stochastic information. The higher the degree of dynamism, the lower is the competitive advantage of the stochastic solution methods.

6 Conclusion

In this book we study the effects of exploiting stochastic information about future circumstances while planning vehicle routes for Austrian ambulance service providers. For this purpose we present three stochastic variants of the dynamic dial-a-ride problem (DDARP). Furthermore, we tailor two conceptually similar pairs of metaheuristic solution methods to the requirements of these problem classes using the well known idea of variable neighborhood search (VNS) as a basis.

In Chapter 2 we present a short description of the basic DDARP which is used as a basis for all three stochastic extensions. This problem is based on a real world road network and consists of planning the vehicle routes for an Austrian ambulance service provider in order to service a given set of (partially) dynamic transportation requests using a limited number of homogeneous vehicles. The passenger ride time is limited by defining a maximum detour restriction of 30 minutes. The objective function used is lexicographic and includes three stages. First, the user inconvenience in terms of the sum of earliness and tardiness with respect to the requests' time windows is to be minimized. Second, the solution costs represented by the number of vehicles used in a solution shall be kept as low as possible. Third, the total mileage of the found solution, which is another cost related aspect, should be minimized.

Based on this basic DDARP, we present the first stochastic extension in Chapter 3. The dynamic stochastic dial-a-ride problem with expected return transports (DSDARP) is based on the assumption that some stochastic information about future return transports (outbound requests) can be exploited during the process of designing the vehicle routes. We adapt two pairs of metaheuristic solution methods to the requirements of the DSDARP in order to study the differences between myopic methods (i.e., which do not take any stochastic information into account while planning) and stochastic methods (i.e., which try to exploit the available stochastic information while planning). To be precise, we present a dynamic variant of the well known VNS methods and a dynamic variant of stochastic VNS (S-VNS) as the first pair. The second pair consists of the multiple plan approach (MPA) as a myopic method and the multiple scenario approach (MSA) as a stochastic variant thereof. This way we can directly compare the results obtained by myopic and stochastic methods and study the effects of exploiting the stochastic information while planning. The results show, that reasonable improvements can be obtained when using stochastic approaches. However, the results reveal a remarkable negative correlation between the return transport probability (which to some extent influences the total degree of dynamism present in a problem instance) and the obtained solution quality.

In Chapter 4 we present the second stochastic extension to the basic DDARP which assumes the availability of stochastic information about future traffic situations. This problem class is denoted as dynamic dial-a-ride problem with stochastic time-dependent travel speeds (DDARPTD). Vehicle speeds along the underlying real world road network are assumed to be time-dependent and additionally influenced by stochastic effects (like, e.g., traffic accidents). We adapt our two pairs of metaheuristics to the requirements of this new problem class and again study the effects of exploiting the relatively limited amount of stochastic information that is available. Additionally, we present an alternative scheduling algorithm for the DDARP with time-dependent travel speeds. Due to the time-dependent nature of the travel speeds, also the objective function needs to be adapted in order to penalize maximum detour violations. The obtained results show that both stochastic methods are able to obtain almost equally large improvements over their myopic counterparts. Again, the total degree of dynamism is negatively correlated with the resulting solution quality.

The third extension to the DDARP is a combination of the DSDARP presented in Chapter 3, the DDARPTD studied in Chapter 4 and additional heterogeneous aspects with respect to the vehicle fleet and the transportation requests. The heterogeneous dynamic stochastic dial-a-ride problem with stochastic time-dependent travel speeds (HDSDARPTD) is based on the assumption that stochastic information about both, future return transports as well as future traffic situations, is available. Additionally, we no longer assume all vehicles an patients to be identical but instead use two different vehicle types with different seating and define three different patient types. In addition to patients who can be transported in a sitting position, some patients require a stretcher or a wheelchair place to be available. Furthermore, some patients are accompanied by an additional person during transportation. Our results show, that exploiting information about both stochastic problem aspects simultaneously leads to more stable improvements than exploiting each of the two stochastic aspects separately. Again, the results indicate that the overall degree of dynamism present in a problem instance ist the most important factor influencing the obtainable improvement.

Summarizing, in this book we present three different stochastic extensions to a real world motivated dynamic dial-a-ride problem as well as four metaheuristic solution approaches tailored to each of these problem classes. The obtained results show that for real world sized ambulance routing problems, the exploitation of available stochastic information during the process of designing the vehicle routes leads to remarkable improvements in solution quality compared to the solutions found by purely myopic methods.

Future research should especially investigate further possibilities of exploiting additional stochastic information about future traffic situations. This will in our opinion be one of the major topics in the near future because the amount of traffic on city roads is steadily increasing. Also re-allocation strategies exploiting the available stochastic information while planning would be an interesting field of research.

List of abbreviations

ARC	Austrian Red Cross
ASBÖ	Arbeiter-Samariter-Bund Österreichs
BSA	block scheduling algorithm
DARP	dial-a-ride problem
DDARP	dynamic dial-a-ride problem
DDARPTD	dynamic dial-a-ride problem with stochastic time-dependent travel speeds
DSDARP	dynamic stochastic dial-a-ride problem
FCD	floating car data
FFG	Austrian Research Promotion Agency
FIFO	first-in-first-out
FWF	Austrian Science Fund
HDARP	heterogeneous dial-a-ride problem
HDSDARPTD	heterogeneous dynamic stochastic dial-a-ride problem with stochastic time-dependent travel speeds
MPA	multiple plan approach
MSA	multiple scenario approach
PTA	patient transport ambulance
S-VNS	stochastic variable neighborhood search
SAE	sample average estimator
VNS	variable neighborhood search
VRPTW	vehicle routing problem with time windows
VSC	Vienna Scientific Cluster

Bibliography

[1] Österreichisches Rotes Kreuz. Tätigkeitsbericht 2009. http://www.roteskreuz.at, May 2010.

[2] Arbeiter-Samariter-Bund Österreichs. Jahresbericht 2009. http://www.samariterbund.net, April 2010.

[3] N.H.M. Wilson and N.J. Colvin. Computer control of the Rochester dial-a-ride system. (R-77-22), 1977.

[4] N.H.M. Wilson and H. Weissberg. Advanced dial-a-ride algorithms research project: Final report. (R76-20), 1976.

[5] N.H.M. Wilson, J.M. Sussman, H.K. Wong, and B.T. Higonnet. Scheduling algorithms for a dial-a-ride system. (USL TR-70-13), 1971.

[6] P. Healy and R. Moll. A new extension of local search applied to the Dial-A-Ride Problem. *European Journal of Operational Research*, 83(1):83–104, 1995.

[7] S.N. Parragh, K.F. Doerner, and R.F. Hartl. A survey on pickup and delivery problems. Part II: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft*, 58(1):81–117, 2008.

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

[9] S.N. Parragh, K.F. Doerner, and R.F. Hartl. Variable Neighborhood Search for the Dial-a-Ride Problem. *Computers & Operations Research*, 37(6):1129–1138, 2010.

[10] J.-F. Cordeau and G. Laporte. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological*, 37(6):579–594, 2003.

[11] H.N. Psaraftis. A dynamic programming solution to the single-vehicle, many-to-many immediate request dial-a-ride problem. *Transportation Science*, 14(2):130–154, 1980.

[12] O.B.G. Madsen, H.F. Ravn, and J.M. Rygaard. A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities and multiple objectives. *Annals of Operations Research*, 60(1):193–208, 1995.

[13] M.E.T Horn. Fleet scheduling and dispatching for demand-responsive passenger services. *Transportation Research Part C*, 10(1):35–63, 2002.

[14] A. Attanasio, J.-F. Cordeau, G. Ghiani, and G. Laporte. Parallel Tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. *Parallel Computing*, 30(3):377–387, 2004.

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

[16] G. Berbeglia, J.-F. Cordeau, and G. Laporte. A Hybrid Tabu Search and Constraint Programming Algorithm for the Dynamic Dial-a-Ride Problem. Technical report, CIRRELT, Université de Montréal, C.P. 6128, succ. Centre-ville, Montréal (Québec), Canada, 2010.

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

[18] M. Schilde, K.F. Doerner, and R.F. Hartl. Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem. Technical report, Department of Business Administration, University of Vienna, Vienna, Austria, 2011.

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

[20] Z. Xiang, C. Chu, and H. Chen. The study of a dynamic dial-a-ride problem under time-dependent and stochastic environments. *European Journal of Operational Research*, 185(2):534–551, 2008.

[21] E. Hyytiä, S. Aalto, A. Penttinen, and R. Sulonen. A stochastic model for a vehicle in a dial-a-ride system. *Operations Research Letters*, 38(5):432–435, 2010.

[22] G. Heilporn, J.-F. Cordeau, and G. Laporte. An integer L-shaped algorithm for the Dial-a-Ride Problem with stochastic customer delays. *Discrete Applied Mathematics*, 159(9):883–895, 2011.

[23] W. Gutjahr, A. Hellmayr, and G. Pflug. Optimal stochastic single-machine tardiness scheduling by stochastic branch-and-bound. *European Journal of Operational Research*, 117(2):396–413, 1999.

[24] G. Laporte, F.V. Louveaux, and L. van Hamme. An Integer L-Shaped Algorithm for the Capacitated Vehicle Routing Problem with Stochastic Demands. *Operations Research*, 50(3):415–423, 2002.

[25] W. Gutjahr. S-ACO: An Ant-Based Approach to Combinatorial Optimization Under Uncertainty. In *4th International Workshop on Ant Colony Optimization and Swarm Intelligence*, pages 238–249. Springer Berlin / Heidelberg, 2004.

[26] P. Jaillet. *Probabilistic Traveling Salesman Problems.* PhD thesis, MIT, Cambridge, Mass., 1985.

[27] D. Bertsimas. *Probabilistic Combinatorial Optimization Problems.* PhD thesis, MIT, Cambridge, Mass., 1988.

[28] D.J. Bertsimas, P. Jaillet, and A.R. Odoni. A priori optimization. *Operations Research*, 38(6):1019–1033, 1990.

[29] G. Laporte, F.V. Louveaux, and H. Mercure. A priori optimization of the probabilistic traveling salesman problem. *Operations Research*, 42(3):543–549, 1994.

[30] D. Teodorović and G. Pavković. A simulated annealing technique approach to the vehicle routing problem in the case of stochastic demand. *Transportation Planning and Technology*, 16(4):261–273, 1992.

[31] M. Gendreau, G. Laporte, and R. Séguin. A Tabu Search Heuristic for the Vehicle Routing Problem with Stochastic Demands and Customers. *Operations Research*, 44(3):469–477, 1996.

[32] B.L. Golden and W.R. Stewart Jr. Vehicle routing with probabilistic demands. In *Computer Science and Statistics: Tenth Annual Symposium on the Interface NBS Special Publication 503*, pages 203–259. 1978.

[33] N. Secomandi and F. Margot. Reoptimization Approaches for the Vehicle-Routing Problem with Stochastic Demands. *Operations Research*, 57(1):214–230, 2009.

[34] F.A. Tillmann. The Multiple Terminal Delivery Problem with Probabilistic Demands. *Transportation Science*, 3(3):192–204, 1969.

[35] G. Clarke and J.W. Wright. Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research*, 12(4):568–581, 1964.

[36] A. Kleywegt, V. Nori, and M. Savelsbergh. The stochastic inventory routing problem with direct deliveries. *Transportation Science*, 36(1):94–118, 2002.

[37] L.M. Hvattum, A. Løkketangen, and G. Laporte. Solving a Dynamic and Stochastic Vehicle Routing Problem with a Sample Scenario Hedging Heuristic. *Transportation Science*, 40(4):421–438, 2006.

[38] W. Gutjahr, S. Katzensteiner, and P. Reiter. A VNS Algorithm for Noisy Problems and its Application to Project Portfolio Analysis. In *4th International Symposium on Stochastic Algorithms: Foundations and Applications*, pages 93–104. Springer Berlin / Heidelberg, 2007.

[39] N. Mladenović and P. Hansen. Variable Neighborhood Search. *Computers and Operations Research*, 24(11):1097–1100, 1997.

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

[41] M. Gendreau, G. Laporte, and R. Séguin. Stochastic vehicle routing. *European Journal of Operational Research*, 88(1):3–12, 1996.

[42] D.J. Rosenkrantz, R.E. Stearns, and P.M. Lewis. Approximate algorithms for the traveling salesperson problem. *IEEE Conference Record of 15th Annual Symposium on Switching and Automata Theory*, pages 33–42, Oct. 1974.

[43] B. Hunsaker and M.W.P. Savelsbergh. Efficient feasibility testing for dial-a-ride problems. *Operations Research Letters*, 30(3):169–173, 2002.

[44] P. Hansen, N. Mladenović, and J.A.M. Peréz. Variable neighborhood search: methods and applications. *Annals of Operations Research*, 175(1):367–407, 2010.

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

[46] S. Kritzinger. Warteschlangentheorie als Instrument zur Simulation von Ambulance Routing. Master's thesis, University of Salzburg, Salzburg, Austria, May 2008.

[47] S. Lorini, J.-Y. Potvin, and N. Zufferey. Online vehicle routing and scheduling with dynamic travel times. *Computers & Operations Research*, 38(7):1086–1090, 2011.

[48] A.L. Kok, E.W. Hans, J.M.J. Schutten, and W.H.M. Zijm. Vehicle Routing with Traffic Congestion and Drivers' Driving and Working Rules. Technical report, Operational Methods for Production and Logistics, University of Twente, P.O. Box 217, 7500AE, Enschede, Netherlands, 2010.

[49] J.-Y. Potvin, Y. Xu, and I. Benyahia. Vehicle routing and scheduling with dynamic travel times. *Computers & Operations Research*, 33(4):1129–1137, 2006.

[50] B. Fleischmann, M. Gietz, and S. Gnutzmann. Time-Varying Travel Times in Vehicle Routing. *Transportation Science*, 38(2):160–173, 2004.

[51] S. Ichoua, M. Gendreau, and J.-Y. Potvin. Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research*, 144(2):379–396, 2003.

[52] W. Maden, R. Eglese, and D. Black. Vehicle routing and scheduling with time-varying data: A case study. *Journal of the Operational Research Society*, 61(3):515–522, 2010.

[53] R. Eglese, W. Maden, and A. Slater. A Road Timetable to aid vehicle routing and scheduling. *Computers & Operations Research*, 33(12):3508–3519, 2006.

[54] B. Fleischmann, S. Gnutzmann, and E. Sandvoß. Dynamic Vehicle Routing Based on Online Traffic Information. *Transportation Science*, 38(4):420–433, 2004.

[55] L. Fu. Improving Paratransit Scheduling by Accounting for Dynamic and Stochastic Variations in Travel Time. *Transportation Research Record*, 1666:74–81, 1999.

[56] L. Fu. Scheduling dial-a-ride paratransit under time-varying, stochastic congestion. *Transportation Research Part B*, 36(6):485–506, 2002.

[57] M. Schneider, M. Linauer, N. Hainitz, and H. Koller. Traveller Information Service based on Real-Time Toll Data in Austria. *IET Intelligent Transportation Systems*, 3(2):124–137, 2009.

[58] M. Linauer. *Generierung streckenbezogener Verkehrsdaten als Basis für den Einsatz in Verkehrstelematiksystemen.* PhD thesis, Institute for Transport Studies, Department of Landscape, Spatial and Infrastructure Sciences, University of Natural Resources and Life Sciences, Vienna, 2005.

[59] P. Laborczi, B. Nowotny, M. Linauer, M. Schneider, and D. Leihs. Optimal route guidance based on floating car data. In *10th World Conference on Transport Research, Istanbul, Turkey*, 2004.

[60] M. Linauer and E. Mrakotsky. Methods to generate Floating Car Data for use in Traffic Telematics Systems. In *11th World Congress and Exhibition on ITS, Nagoya, Japan*, 2004.

[61] B.-H. Ahn and J.-Y. Shin. Vehicle-Routeing with Time Windows and Time-Varying Congestion. *The Journal of the Operational Research Society*, 42(5):393–400, 1991.

[62] Savelsbergh M.W.P. The Vehicle Routing Problem with Time Windows: Minimizing Route Duration. *ORSA Journal on Computing*, 4(2):146–154, 1992.

[63] P. Toth and D. Vigo. Heuristic Algorithms for the Handicapped Persons Transportation Problem. *Transportation Science*, 31(1):60–71, 1997.

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

[65] B. Rekiek, A. Delchambre, and H.A. Saleh. Handicapped Person Transportation: An application of the Grouping Genetic Algorithm. *Engineering Applications of Artificial Intelligence*, 19(5):511–520, 2006.

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

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

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

[69] S.N. Parragh. 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, 2011.

Abstract

The field of research regarding the optimization of humanitarian aid as well as health care efforts is very wide. During the last decades, the efforts expended in research projects related to this area are steadily increasing. In this book, we address a specific problem out of this large field: the transportation of elderly, ill and disabled persons. Such problems are commonly referred to as dial-a-ride problem (DARP) in the literature.

We study three dynamic and stochastic variants of this problem type with the aim of examining the effects of exploiting stochastic information about different future circumstances while planning. First, we consider stochastic information about future return transports and tailor two pairs of metaheuristic solution methods to the requirements of this problem. Second, we study the usage of stochastic information about future travel speeds while constructing the vehicle routes. Finally, we combine these two stochastic aspects with additional heterogeneous extensions regarding the vehicle fleet, the transported patients and multiple depots.

Based on our findings we identify factors which have a strong influence on the potential benefits of exploiting stochastic information about future circumstances. Generally speaking, the benefits obtainable by planning in a stochastic way can be remarkable if the underlying conditions are suitable. Especially the total degree of dynamism present in a problem setting turns out to be negatively correlated with the achieved solution quality.

Zusammenfassung

Die wissenschaftliche Forschung in Bezug auf humanitäre Hilfe sowie Gesundheitsfürsorge ist sehr weitläufig. Im Laufe der letzten Jahrzehnte stiegen die Anstrengungen welche in einschlägige Forschungsprojekte investiert wurden stetig an.

In diesem Buch betrachten wir einen konkreten Problemfall aus diesem großen Forschungsbereich: den Transport von alten, kranken oder in ihrer Mobilität eingeschränkten Personen. Probleme dieser Art werden in der Fachliteratur regelmäßig als *dial-a-ride* Probleme bezeichnet.

Wir studieren drei dynamisch und stochastische Varianten dieser Problemklasse mit dem Ziel, Effekte, welche durch das Ausnutzen von stochastischen Informationen über zukünftige Umstände während der Planung verursacht werden, zu untersuchen. Zunächst betrachten wir stochastische Informationen über zukünftige Rücktransporte und adaptieren zwei Paare von metaheuristischen Lösungsverfahren für diese Problemstellung. Anschließend untersuchen wir die Ausnutzung stochastischer Informationen über zukünftige Verkehrsbedingungen während der Planung der Fahrzeugrouten. Schlussendlich kombinieren wir diese beiden stochastischen Einflüsse mit zusätzlichen Aspekten bezüglich heterogener Fahrzeugflotten, heterogener Patienten sowie mehrerer Heimatstandorte.

Basierend auf unseren Ergebnissen identifizieren wir Faktoren, welche starken Einfluss auf das mögliche Verbesserungspotential haben, das durch Ausnutzung der stochastischen Informationen erreichbar ist. Grundsätzlich sind die Vorteile, die durch eine stochastische Planung erreicht werden können bemerkenswert, sofern die entsprechenden Rahmenbedingungen vorliegen. Insbesondere das in einer Problemstellung vorhandene Ausmaß an Dynamik erweist sich als mit der erreichten Lösungsqualität stark negativ korrelierend.

Curriculum vitae

Personal Data

name	Michael Schilde
date of birth	January 27, 1983
citizenship	Austria

Education

2007–present	*PhD Program in Management*, University of Vienna, Vienna, Austria
August 2009	*PhD course on Applications of Vehicle Routing*, Aarhus School of Business, Aarhus University, Aarhus, Denmark
2003–2007	*Master Program in International Business Administration*, University of Vienna, Vienna, Austria. (Academic degree: mag. rer. soc. oec.)

Professional Activities

2008–present	*Research project assistant*, University of Vienna, Department of Business Administration, Chair of Production and Operations Management, Vienna, Austria
2007–2008	*Tutor*, University of Vienna, Department of Business Administration, Chair of Production and Operations Management, Vienna, Austria

Publications

2011 M. Schilde, K.F. Doerner and R.F. Hartl. *Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem*. Technical report, University of Vienna, 2011. Submitted.

M. Schilde, K.F. Doerner and R.F. Hartl. *Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports*. Computers & Operations Research 38(2): pp. 1719–1730, 2011.

E. Reschenhofer, M. Schilde, E. Oberecker, E. Payr, H. Tandogan and L. Wakolbinger. *Identifying the determinants of foreign direct investment: a data-specific model selection approach*. Statistical Papers, published online first: 26 February 2011, DOI: 10.1007/s00362-011-0377-2, 2011.

2009 M. Schilde, K.F. Doerner, R.F. Hartl and G. Kiechle. *Metaheuristics for the Bi-Objective Orienteering Problem*. Swarm intelligence 3(3): pp. 179–201, 2009.

Talks and Presentations

2011	M. Schilde, K.F. Doerner and R.F. Hartl. *Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem*. EU/MEeting 2011, Vienna, Austria, February 2011.
2010	M. Schilde, K.F. Doerner and R.F. Hartl. *Solution Methods for the Dynamic Dial-a-Ride Problem with Stochastic Time-Dependent Travel Times*. International Conference Operations Research 2010, Munich, Germany, September 2010.
	M. Schilde, K.F. Doerner and R.F. Hartl. *Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports*. EURO XXIV, Lisbon, Portugal, July 2010.
	M. Schilde, K.F. Doerner and R.F. Hartl. *Solution methods for the dynamic dial-a-ride problem with stochastic requests and expected return trips*. TRISTAN VII, Troms, Norway, June 2010.
	M. Schilde, K.F. Doerner and R.F. Hartl. *Metaheuristiken für das dynamisch stochastische Dial-a-Ride Problem mit erwarteten Rücktransporten*. Fachtagung: Entscheidungsunterstützung in der Logistik - Geographische Informationssysteme und Optimierung, Salzburg, Austria, April 2010.
2009	M. Schilde, K.F. Doerner and R.F. Hartl. *Solution methods for the dynamic dial-a-ride problem with stochastic requests and expected return trips*. MIC 2009, Hamburg, Germany, July 2009.
	M. Schilde, K.F. Doerner and R.F. Hartl. *Metaheuristics for the Bi-Objective Orienteering Problem*. EU/MEeting 2009, Porto, Portugal, April 2009.

Awards and Prices

2007	*Bank Austria Price for exceptional diploma thesis in Operations Research*. OeGOR Special Award.
2007	*First place in "Best of the Best" ranking for International Business Administration 2006/2007*. Sponsored by UniPort and Raiffeisen Zentralbank.
2002	*Thesis price: "TimePro Network based time recording system"*. IT-Contest sponsored by the Austrian Chamber of Labor.

Teaching

since 2008	*Production and logistics*, University of Vienna. Content: newsboy problems, master production schedule, material requirements planning, TSP, etc.