Computers & Operations Research 38 (2011) 1719–1730

Contents lists available at ScienceDirect

Computers & Operations Research

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

Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports

M. Schilde [a,*], K.F. Doerner [a,b], R.F. Hartl [a]

[a] University of Vienna, Department of Business Administration, Bruenner Strasse 72, 1210 Vienna, Austria
[b] Johannes Kepler University Linz, Department of Production and Logistics, Altenberger Strasse 69, 4040 Linz, Austria

ARTICLE INFO

Available online 17 February 2011

Keywords:
Transportation
Dial-a-ride problem
Dynamic problem
Stochastic information
Metaheuristic
Variable neighborhood search
Multiple plan approach
Multiple scenario approach
Stochastic variable neighborhood search

ABSTRACT

The problem of transporting patients or elderly people has been widely studied in literature and is usually modeled as a dial-a-ride problem (DARP). In this paper we analyze the corresponding problem arising in the daily operation of the Austrian Red Cross. This nongovernmental organization is the largest organization performing patient transportation in Austria. The aim is to design vehicle routes to serve partially dynamic transportation requests using a fixed vehicle fleet. Each request requires transportation from a patient's home location to a hospital (outbound request) or back home from the hospital (inbound request). Some of these requests are known in advance. Some requests are dynamic in the sense that they appear during the day without any prior information. Finally, some inbound requests are stochastic. More precisely, with a certain probability each outbound request causes a corresponding inbound request on the same day. Some stochastic information about these return transports is available from historical data. The purpose of this study is to investigate, whether using this information in designing the routes has a significant positive effect on the solution quality. The problem is modeled as a dynamic stochastic dial-a-ride problem with expected return transports. We propose four different modifications of metaheuristic solution approaches for this problem. In detail, we test dynamic versions of variable neighborhood search (VNS) and stochastic VNS (S-VNS) as well as modified versions of the multiple plan approach (MPA) and the multiple scenario approach (MSA). Tests are performed using 12 sets of test instances based on a real road network. Various demand scenarios are generated based on the available real data. Results show that using the stochastic information on return transports leads to average improvements of around 15%. Moreover, improvements of up to 41% can be achieved for some test instances.

1. Introduction

This paper was motivated by the problem the Austrian Red Cross (ARC) faces in its daily operation. This nongovernmental organization is responsible for performing most of the patient transportation tasks arising in Austria. Patients can call the ARC to arrange so called "taxi transports". Such transports start at the patient's home location and end at a specific hospital (outbound requests). For each request, a time window for arrival at the hospital is specified by the patient. Subsequently, the ARC determines a time window for departure from the patient's origin location. This is communicated to the patient to be prepared for departure. In addition, hospitals may arrange transports for patients that need to be transported back home after treatment (inbound request). Some of the requests are known in advance (i.e., the patient or hospital arranged the transport the previous day or earlier), whereas others arise as the day progresses.

All these transportation requests have to be served by a fixed vehicle fleet. All vehicles are based at a common depot location. No vehicle can transport more than three patients at the same time. In this paper we consider only homogenous transportation requests. This means that patients are assumed to occupy exactly one seat in a vehicle and we do not distinguish between different modes of transportation (e.g., patient seat, stretcher, wheelchair). As customer satisfaction is very important for the ARC, the rejection of transportation requests is not allowed. Additionally, no detour of more than 30 min can be planned while a patient is aboard (e.g., for picking up or dropping another patient). For the same reason, violating time windows shall be avoided whenever possible. However, time windows are soft and can be violated if absolutely necessary. The ARC explains this as follows. If a patient is delivered to the hospital after the specified time window, the patient's assignment to specialized equipment (e.g., a dialysis

* Corresponding author.
E-mail addresses: michael.schilde@univie.ac.at (M. Schilde), karl.doerner@univie.ac.at (K.F. Doerner), richard.hartl@univie.ac.at (R.F. Hartl).

0305-0548
doi:10.1016/j.cor.2011.02.006

machine) may become obsolete. This forces the patient and hospital to re-schedule the treatment which decreases customer satisfaction. Economic aspects like the number of vehicles used or the total distance traveled are only regarded as secondary objectives. The ARC aims at designing vehicle routes that optimize these objectives.

A similar problem also arises in everyday business of many other organizations in the field of passenger transportation like, for example, cab companies. However, our problem includes an additional aspect. Stochastic information about expected return transports is available from historic data. Each patient that is transported to a hospital might, at a later point in time, require transportation back to his home location. These return transports can be caused by both static and dynamic outbound requests. Return transports themselves are always dynamic. Even if the patient knows about the return transport in advance (e.g., in case of a dialysis treatment), this information is not available to the ARC. Nevertheless, this information can be estimated quite well using the available historical data.

Imagine a person is regularly transported from home to a hospital for dialysis and back home afterwards. Such a treatment normally consumes a predictable amount of time. This is independent of who the person is or when the treatment takes place. Thus, it is possible to determine the parameters for the statistical distribution of such return transports. This information can then be used when designing the vehicle routes. More precisely, the stochastic information can be generated from historical data about treatment durations. The same procedure would of course make sense for every type of return transport if the required information is available. What makes this problem unique is that for possible return transports, such information can be used for planning. Note that we also studied the case in which information about the dynamic transportation requests from a patients' home location to the hospital is assumed to be available, but including this additional information brings only limited benefit (see Section 5.3 for details).

The problem of point-to-point passenger transportation is commonly modeled as a dial-a-ride problem (DARP) in the literature (see, e.g., [1,2] for recent surveys). Healy and Moll [3] showed that the DARP is NP-hard. Thus, much effort has been spent in finding efficient and effective solution methods for this problem class. The static deterministic variants of the DARP are well studied and very sophisticated solution approaches were presented within the last years (see, e.g., [4,5] for state-of-the-art metaheuristics). Also, dynamic variants of this problem have lately received some attention (see, e.g., [6]). Xiang et al. [7] recently studied a dynamic stochastic variant of the DARP. They proposed a simple local search heuristic that uses a secondary objective function to escape from local optima.

Several studies incorporate some kind of stochastic information in the context of other problems. Laporte and Louveaux [8] proposed an integer L-shaped method for stochastic integer problems with complete recourse. Gutjahr et al. [9] developed a stochastic branch-and-bound method for optimal single-machine tardiness scheduling. Gutjahr [10] also proposed an ant-based approach to combinatorial optimization problems under uncertainty. Jaillet [11], Bertsimas [12], Bertsimas et al. [13] and Laporte et al. [14] studied probabilistic versions of the traveling salesman problem and other combinatorial optimization problems. Teodorović and Pavković [15], Gendreau et al. [16], and Golden and Stewart [17] published several solution methods for stochastic variants of the vehicle routing problem. Secomandi and Margot [18] proposed re-optimization approaches for the vehicle routing problem with stochastic demands. Tillmann [19] proposed a modification of the well known Clarke and Wright [20] savings algorithm for the multiple terminal delivery problem with probabilistic demands. Kleywegt et al. [21] 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. [22]. Gutjahr et al. [23] recently introduced a stochastic variant of the well known variable neighborhood search (VNS, [24]) 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 [25] 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. MSA additionally takes into account scenarios of the future while planning.

There is much literature available on stochastic methods for different applications. However, to the best of our knowledge, using stochastic information about future transportation requests for solving the dynamic stochastic DARP (DSDARP) was not studied so far. The contribution of our paper is thus threefold:

- We adapt four different metaheuristic methods to the special requirements of the DSDARP (VNS, S-VNS, MPA, and MSA) and identify problem characteristics for which one or the other method works better.
- We determine influence factors for the successful use of stochastic information to improve solution quality.
- We study how far ahead information about the future should be integrated into the solution process.

The remainder of this paper is structured as follows. Section 2 gives a more formal verbal description of the problem at hand. A detailed description of the used solution approaches is given in Section 3, followed by an overview and description of the used test instances in Section 4. A discussion of the parameter settings and obtained results is given in Section 5. The paper concludes with a short summary and an outlook on future research in Section 6.

2. Problem description

We modeled the problem at hand as a DSDARP on a (directed) real world road network. Each transportation request r consists of two separate nodes (p_r, d_r) representing the pickup and delivery location, respectively. Each node n is assigned a quantity of $q_n = 1$ for pickup locations and $q_n = -1$ for delivery locations, indicating that one patient is boarding or leaving the vehicle. The depot node 0 is assigned a quantity of $q_0 = 0$. Every vehicle has a limited capacity of $Q=3$. This means that vehicle fleet and customers are assumed to be homogenous in our model. Variants of the static DARP with heterogeneous patients and heterogeneous fleet were studied in the past by Parragh et al. [26,27].

Each node n (pickup or delivery) has a time window $[e_n, l_n]$. The depot node 0 has the time window $[0, T_{max}]$. Vehicles can leave the depot at time $e_0 = 0$ and should not return later than time $l_0 = T_{max}$. Here, T_{max} is the duration of the working shift of the vehicle crew. Arriving at the depot later than l_0 causes overtime payments and should therefore be avoided. Service at a location must not start before the corresponding time window. Early arrivals would lead to waiting times. The end of each time window is modeled as being soft. If service starts after the end of the time window this difference is denoted as tardiness and is penalized in the objective function. A late return to the depot is penalized in the same way. This way we make sure that every

request can be inserted feasibly into any solution at any time. Therefore, we never have to reject incoming requests.

A DARP usually considers user inconvenience in the objective or as constraints. In our case, following the requirement specified by the Austrian Red Cross, we imposed a maximum detour constraint of 30 min in the following sense. Let the time required to go directly from p_r to d_r be t_{direct}. 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 $t_{real} \leq t_{direct} + 30$ must not be violated.

We assumed that transportation request r arises at time a_r. This information represents the moment in time at which the ARC is informed about the transportation request (usually by phone). Some of the requests (static requests) are known in advance, i.e., they arise at time $a_r = 0$. Some requests are dynamic in the sense that they appear during the day, $a_r > 0$. Both static and dynamic outbound requests can cause return (inbound) transports, which are stochastic. More precisely, with a certain probability each outbound request causes a corresponding inbound request on the same day. Some stochastic information about the occurrence and timing of these return transports is available from historical data. See Section 4.1 for more details.

We used a lexicographic objective function. The primary objective is to minimize the total tardiness 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 have the same total tardiness, the secondary objective is used for comparison. Only if both, primary and secondary objective, are equal for two solutions, comparison is based on the third objective.

3. Solution methods

The primary aim of this paper was to study whether using stochastic information about future return transports has a beneficial effect on solution quality of the DSDARP. We investigated four different approaches that are all based on metaheuristic methods for the static variant. We based our analysis on metaheuristic approaches which are as similar as possible in design. The reason is that we wanted to guarantee that differences in solution quality are caused by the fact whether or not and how the additional information was used in the optimization and not by conceptual differences between the underlying optimization methods. Parragh et al. [4] developed an efficient VNS approach for the static DARP. The neighborhood operators used were well tested and documented. Therefore, we decided to use this VNS concept as the basis for our modifications.

First, we adapted this VNS to the requirements of the dynamic stochastic DARP, resulting in a dynamic VNS approach. In this basic variant, the stochastic information is ignored and the stochastic requests are simply treated as dynamic ones. Second, we extended this dynamic VNS to a dynamic S-VNS method by incorporating available stochastic information along the lines of the S-VNS concept reported in the literature [23]. Third, we applied the MPA and MSA described by Bent and Van Hentenryck [25] and developed some modifications that worked better. Therefore, using two different basic approaches (VNS, S-VNS) and (MPA, MSA) we could investigate the potential benefits of using stochastic information while planning (S-VNS, MSA) compared to ignoring this information (VNS, MPA).

The following subsections present detailed information about the implementation of the simulation framework as well as the solution approaches. All design decisions presented in the following subsections are the result of extensive (pre-)testing. What is presented here is the most successful setting for each of the methods according to these tests.

3.1. Simulation framework

As the problem at hand is dynamic, a simulation framework was required to handle the management of transportation requests during execution. This framework was designed to run in the background, handling the problem specific information (test instance, distance matrix, vehicle fleet, etc.) and keeping the solver modules informed about arising transportation requests during execution. Also, this framework is in charge of managing simulation time.

After loading the problem data, the simulator starts the solver module and initializes simulation time to 0. Whenever a solution approach reaches a point in execution at which it could possibly handle new transportation requests, it sends an update request to the simulator. The simulator then updates simulation time proportional to the actual CPU time elapsed since starting the solver module. After that, it delivers a list of transportation requests that arose since the last update request to the solver module. Proportional hereby means that simulation time does not necessarily equal run time. Nevertheless, all of our test runs were performed in real time.

3.2. Dynamic VNS

In Algorithm 1 we briefly summarize the traditional structure of a VNS as developed for static optimization problems in [24]. This structure requires some adaptation if used to solve dynamic problems. An outline of this modification, the dynamic VNS algorithm, is given in Algorithm 2. The main difference to the static version of VNS can be seen on line 4. Here, the dynamic nature of the problem requires the algorithm to periodically check whether new transportation requests have occurred during execution.

The shaking operators used (see below) already implicitly include some kind of local search behavior. We tested whether an additional local search step as in the traditional VNS is beneficial for solution quality. As the results did not show a significant (positive) effect on solution quality, we decided to use a reduced VNS (without local search) to keep computation time short. Thus, line 5 of the static version was omitted in our dynamic version.

As our test instances each represent one working day at the ARC, the run time of our solution methods is the only stopping criterion. If the end of the simulated day (which equals 10 h of run time) was reached, dynamic VNS was stopped and the resulting solution details were saved.

Algorithm 1. Structure of traditional VNS [24].

```
1: x ← InitialSolution()
2: N(κ) ← SelectFirstNeighborhood()
3: while StoppingCriterionNotMet() do
4:     x' ← ShakeSolution(x,N(κ))
5:     x'' ← LocalSearch(x'')
6:     x ← MoveOrNot(x,x'')
7:     N(κ) ← SelectNextNeighborhood(κ)
8: end while
9: return x as best found solution
```

Algorithm 2. Structure of dynamic VNS.

```
1: x ← InitialSolution()
2: N(κ) ← SelectFirstNeighborhood()
3: while StoppingCriterionNotMet() do
4:     x ← InsertNewRequests(x)
5:     x' ← ShakeSolution(x,N(κ))
6:     x ← MoveOrNot(x,x')
7:     N(κ) ← SelectNextNeighborhood(κ)
8: end while
9: return x as best found solution
```

3.2.1. Initial solution

The initial solution used in our algorithm was generated using a modified version of the cheapest insertion heuristic reported by Rosenkrantz et al. [28] for the traveling salesman problem. The outline of this modified method is given in Algorithm 3. In the pseudocode, we use the relation $x \gg y$ to indicate that x is better than y. The method starts by obtaining a list of static transportation requests from the simulator and initializes the (partial) solution with only one empty route. The algorithm then iteratively inserts the requests from this list into the solution one after the other. When inserting a request, the algorithm checks all feasible combinations of insertion positions for the pickup and delivery service in every route. Out of these positions, the algorithm selects the one that leads to the lowest deterioration in solution quality. The method also considers inserting the current request into a new empty route, if not all vehicles are already in use. This option is selected only if the increase in tardiness caused by an insertion into an existing route would exceed a threshold level τ which is an input parameter to the method.

The feasibility of any pair of insertion positions for pickup and delivery service is checked using a modified version of the scheduling algorithm reported by Hunsaker and Savelsbergh [29] with a reported algorithmic complexity of $\mathcal{O}(n)$. This method had to be adapted such that time windows are treated as soft (including the one for the depot at the end of day). Additionally, all services that have already been performed and services to which a vehicle has already departed are frozen and must not be altered by the algorithm. 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.

Algorithm 3. Modified cheapest insertion heuristic.

```
1:  R ← ListOfKnownRequests()
2:  x ← AddEmptyRoute()
3:  for r ∈ R do
4:      x' ← InsertRequestAtBestPosition(x, r)
5:      x'' ← InsertRequestIntoNewRoute(x, r)
6:      if x' ≫ x'' then
7:          x ← x'
8:      else
9:          x ← x''
10:     end if
11: end for
12: return x as best found solution
```

3.2.2. Insert new requests

During execution, dynamic VNS periodically checks for new transportation requests. If such requests occurred since the last check, they need to be inserted into the current (partial) solution, as requests must never be rejected in our problem setting. This insertion is performed using the same cheapest insertion method as for the initial solution. The only difference is that only insertion positions after the last service to which a vehicle has already departed are feasible.

3.2.3. Shake solution

Since Parragh et al. [4] proposed a set of highly efficient and effective neighborhood structures for the static DARP we decided to use the same. However, because of the dynamic nature of our problem, these operators required some slight modifications. As with the insertion of new requests, already (partly) serviced transportation requests and requests to which a vehicle had already departed must not be re-scheduled by any operation. The neighborhood structure we used consists of four neighborhood operators: *move*, *swap*, *chain*, and *zero split*. Each of these operators was applied using five intensity levels $\kappa = \{1 \dots 5\}$.

The move operator randomly selects one route and removes κ randomly selected transportation requests from it. Always both services (pickup and delivery) have to be removed. These requests are then iteratively re-inserted into the solution in the route and at the position where they fit best, using the same insertion procedure as described before.

The swap operator randomly selects two routes and removes a randomly selected sequence of up to κ consecutive requests from each of them. These removed requests are then iteratively re-inserted into the other route where they fit best.

The chain operator randomly selects one route as origin route and another one as destination route. It then removes a sequence of up to κ consecutive request from the origin route and iteratively re-inserts them into the destination route at the best position, respectively. This is repeated κ times using the preceding step's destination route as new origin route and another randomly selected destination route.

The zero split operator randomly selects one route and determines all positions in the sequence of services at which no person is currently on board of this vehicle ("zero split points"). The operator then randomly selects two of these points, whereby up to $\kappa-1$ other zero split points may be in between them. These two zero split points delimit a sequence of services that is removed from the route. Afterwards, these requests are iteratively re-inserted into the solution in the route and at the position where they fit best.

3.2.4. Time complexity of neighborhood operators

Let $\overline{x} = \{x_1, x_2, x_3, \dots, x_v\}$ be a solution consisting of v 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 $\eta^2/2$ insertion positions for both, the pickup and delivery part of each removed request in each of the v 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 $\kappa v \eta^3/2$ 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 $2\kappa\eta^3/2$ 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 $\kappa^2\eta^3/2$ 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 $\eta^2/2$ insertion positions in every route, the computational effort is $v\eta^4/2$ 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.

3.2.5. Move or not and select next neighborhood

In the move or not step, we do accept a candidate solution as new current incumbent solution if and only if it is better than the current incumbent. Infeasible solutions (violating capacities or patient ride times) are never accepted. If a candidate solution was not accepted as new current incumbent solution, the function *SelectNextNeighborhood* increases the intensity level κ for the currently used neighborhood operator by 1. If the new level exceeds the maximum intensity of 5, it selects the next neighborhood operator with the lowest intensity $\kappa = 1$. If the last neighborhood operator was already used, the first neighborhood operator is selected again. The sequence in which neighborhood operators are selected is as follows: move → swap → chain → zero split.

3.3. Dynamic S-VNS

The outline of the traditional S-VNS method (without dynamic requests) is shown in Algorithm 4. As in traditional VNS, the algorithm starts by constructing an initial solution. After that, it iteratively searches for better solutions and ends if some stopping criterion is met. The main difference between traditional VNS and S-VNS is found in the comparison of solutions to each other. Traditional VNS ignores any possible information about the future, taking only certain information into account. In contrast to that, S-VNS uses available stochastic information in this comparison process. This difference can be seen in lines 11, 19, and 23 of Algorithm 4. Whenever a comparison needs to be performed, the algorithm samples a set of s_0 scenarios of possible future return transports. More precisely, each of these s_0 scenarios consists of a certain number of return transports that are generated according to the stochastic information. Then these return transports are inserted into the candidate solution one after another using the insertion algorithm described before. The sample average estimator (SAE) for the expected objective function value is simply the average over all s_0 scenarios. All solutions are compared to each other based on this SAE value. This way, the solution providing the better average performance with respect to the used set of sampled return transports is preferred. Note that the SAE is also used in determining the "best" neighbor in line 10 of Algorithm 4.

In addition to the current incumbent solution x, S-VNS also keeps track of the best solution found so far $\hat{x}$. This solution is evaluated against the current incumbent solution using a set of s_m scenarios in a *Tournament* step at each iteration (line 23 in Algorithm 4). This means that the scenarios used to compare the current incumbent solution to the best so far solution are different from the ones used to compare the current incumbent solution to a candidate solution. By continuously increasing the size s_m of this set, the algorithm tends to prefer more robust solutions toward the end of the search process.

A straight forward modification of this algorithm to the requirements of dynamic stochastic DARP is given in Algorithm 5 using a more compact notation. For instance, the (stochastic) local search of lines 7–17 in Algorithm 4 is condensed to line 9 here. To cope with the dynamic nature of the problem, new requests are inserted in the current incumbent and best so far solution in steps 6 and 7 similar to dynamic VNS (line 4 in Algorithm 2).

Unfortunately, evaluating a solution with regard to a set of sampled future return transports is very time consuming in the context of the DSDARP. As the required number of such evaluations is very high in the straight forward modification of S-VNS (Algorithm 5), this version cannot be used to efficiently solve the problem at hand. For this reason and to keep the conceptual difference between dynamic VNS and our dynamic version of S-VNS as small as possible, we integrated further modifications into the concept of S-VNS. The resulting condensed version, which we call dynamic S-VNS, is presented in Algorithm 6. This method uses only the current incumbent solution, thus omitting the best so far solution used in traditional S-VNS. This way, also the *Tournament* step including all its required comparisons is avoided. Furthermore, the method always uses only one sample of future return transports when comparing two solutions in the *MoveOrNotSAE* step ($s_0 = 1$). As already observed with our dynamic VNS method, the *LocalSearch* step does not lead to significantly better solution quality and is therefore omitted as well.

In addition to the modifications already mentioned, we observed that the time horizon, for which samples of the future are taken into account, does also play an important role. The parameter S_{max} in line 6 of Algorithm 6 defines this sampling horizon. This means that possible future return transports are only taken into account in the sample if their pickup time window will start not more than S_{max} minutes in the future. This way, the number of sampled transportation requests can be reduced drastically, thus leading to less time consumption when comparing solutions. Using a smaller value for S_{max}, the algorithm will be able to perform more evaluations in the same amount of time which might lead to better results. However, a small value for S_{max} bears the risk of ignoring too much information about possible future transportation requests, thus leading to inferior results. Therefore, tuning this parameter should be performed carefully. A summary of tested values and resulting changes in solution quality is given in Section 5.

Note that setting S_{max} to a value of 0 does not reduce dynamic S-VNS to dynamic VNS as one would think at first glance. This is caused by the fact that dynamic S-VNS samples return transports for all known requests if their return transport did not yet arise. This way, using a value of $S_{max} = 0$ may still bring up possible return transports (with pickup time window starting now) that, according to the underlying distribution, could have already arisen in the past, but might still arise. Tests showed that even using only this stochastic information about the past may be beneficial in some cases.

Algorithm 4. Traditional S-VNS [23].

```
1:  x ← InitialSolution()
2:  x̂ ← x
2:  m ← 1
4:  N(κ) ← SelectFirstNeighborhood()
5:  while StoppingCriterionNotMet() do
6:      x′ ← ShakeSolution(x,N(κ))
7:      ρ ← 1
8:      repeat
9:          Z ← SampleFutureRequests(s0)
10:         x″ ← BestNeighbor(x′,Z)
11:         if SAE(x″,Z) ≫ SAE(x′,Z) then
12:             x′ ← x″
13:             ρ ← ρ + 1
14:         else
15:             ρ ← ρmax + 1
16:         end if
17:     until ρ > ρmax
18:     Z′ ← SampleFutureRequests(s0)
19:     if SAE(x′,Z′) ≫ SAE(x,Z′) then
20:         x ← x′
21:     end if
22:     Z″ ← SampleFutureRequests(sm)
23:     if SAE(x,Z″) ≫ SAE(x̂,Z″) then
24:         x̂ ← x
25:     end if
26:     N(κ) ← SelectNextNeighborhood(κ)
27:     m ← SelectNextTournamentSampleSize(m)
28: end while
29: return x̂ as best found solution
```

Algorithm 5. 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(κ)
13: $m \leftarrow$ SelectNextTournamentSampleSize(m)
14: **end while**
15: **return** $\hat{x}$ as best found solution

Algorithm 6. Dynamic S-VNS.

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(κ)
9: **end while**
10: **return** x as best found solution

3.4. MPA and MSA

In addition to dynamic VNS and dynamic S-VNS, we tested the multiple plan approach (MPA) and multiple scenario approach (MSA) proposed by Bent and Van Hentenryck [25]. These two methods were already designed for the dynamic vehicle routing problem with time windows (VRPTW) and therefore we expected that not as much adaptation would be required as for the dynamic S-VNS. According to Bent and Van Hentenryck, the two frameworks are designed to work not only with large neighborhood search, which is the local search component originally used, but also with any other local search method as well. We therefore decided to use our dynamic VNS implementation as search component to keep the differences between the four methods as small as possible and therefore obtain comparable results.

The main component of MPA is a pool of solutions which serves as long term memory storing all previously found solutions to the problem. This means that whenever the local search component (which is continuously searching for new solutions) encounters a solution that was not found before, it stores this solution in the pool regardless of solution quality. This idea is based on the fact that a solution that seems to be of high quality at the moment might, in the future, turn out to be a very bad solution because of additional transportation requests that arise dynamically. At any point in time the algorithm needs to select one of the solutions out of this pool as a current incumbent solution that is actually executed by the vehicle fleet. According to Bent and Van Hentenryck [25], this selection should be performed using a consensus function to select the solution that is most similar to all other solutions. Additionally, the pool needs to be updated repeatedly to ensure compatibility between all solutions up to the current point in time. These updates are triggered by four event types. A "timeout event" occurs whenever a vehicle should depart in some solution but should still wait according to the current incumbent solution. A "departure event" occurs whenever a vehicle should depart according to the current incumbent solution but still waits in some other solution. A "new request event" is caused by the arrival of a new transportation request. A "new solution event" occurs whenever the local search component finds a new solution.

An outline of the MPA structure is shown in Algorithm 7. During each iteration, the algorithm selects one solution out of the pool as a current incumbent solution (line 4). It then removes all solutions from the pool that are incompatible with the decisions made so far (line 6/7). This happens if a vehicle should have departed according to the current incumbent solution, but did not in another solution (timeout) or vice versa (departure). If a solution is compatible with the current incumbent solution, newly available requests are inserted into it (line 9). Note that we are always able to insert any new request into any solution because of the soft time windows. Next, the shaking and local search procedures are applied to the current solution x (lines 12 and 13). Finally, the resulting solution is stored in the pool if it was not found before (line 15) and the neighborhood operator for the next iteration is selected accordingly (line 17). The selection of neighborhoods is based on the same order as used for the dynamic VNS described before (move → swap → chain → zero split).

Algorithm 7. Structure of MPA [25].

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

Similar to traditional VNS, MPA ignores any possible information about the future, taking only certain information into account. Therefore, MSA is a logical extension to the MPA concept, introducing the use of stochastic information in the solution process. Other than (dynamic) S-VNS, this is not done by evaluating solutions based on sampled future requests when comparing two solutions. Instead, the starting solution used within the local search component is extended by including sampled future requests. The local search component then continuously tries to find better solutions to this scenario of the future. Then, the sampled requests are discarded from the found solution again. By doing so, the algorithm tries to produce gaps in the schedule of requests to ensure easy integration of arising future transportation request. An outline of the MSA is given in Algorithm 8.

Algorithm 8. Structure of MSA [25].

1: $P \leftarrow$ InitialSolution()
2: $\mathcal{N}(\kappa) \leftarrow$ SelectFirstNeighborhood()
3: **while** StoppingCriterionNotMet() **do**
4: $x \leftarrow$ SelectCurrentIncumbent(P)
5: **for** $\overline{x} \in P$ **do**

```
6:            if (x̄ ≠ x) ∧ (Timeout(x̄,x) ∨ Departure(x̄,x))
              then
7:                P ← P\{x̄}
8:            else
9:                InsertNewRequests(x̄)

10:           end if
11:       end for
12:       x′ ← AddSampledRequestsToSolution(x)
13:       x″ ← ShakeSolution(x′,N(κ))
14:       x‴ ← LocalSearch(x″)
15:       x⁗ ← RemoveSampledRequestsFromSolution(x‴)
16:       if x⁗ ∉ P then
17:           P ← P ∪ {x⁗}
18:       end if
19:       N(κ) ← SelectNextNeighborhood(κ)
20:   end while
```

Surprisingly, our results showed that, with respect to our specific problem, MPA and MSA can perform better when altering the originally proposed design [25]. Both methods lead to slightly better results when using the best solution in the pool as current incumbent solution instead of the one selected by the proposed consensus function. Additionally, always using this current incumbent solution as starting point for the local search component was preferable. Furthermore, we found out that the method showed a random search like behavior when using our dynamic VNS as local search component. This seems to be caused by over diversification in the search process. We assume that this was induced by combining the pool of solutions with our strong neighborhood operators. Therefore, we used only the move neighborhood with $\kappa \in \{1 \dots 5\}$ instead of all four neighborhood operators. Also, for these two methods, using an additional local search step for intensification showed a positive effect on solution quality. Using these modifications, both methods were able to come up with competitive results when compared to the dynamic VNS. We were able to verify the expected effect that MSA returns significantly better results than MPA. However, the improvement obtained is not as large as the one dynamic S-VNS can obtain compared to dynamic VNS. This shows that the effect of incorporating stochastic information while planning indeed is depending on the used solution approach. Detailed results are given in Section 5.

4. Computational experiments

For the computational experiments performed, we generated a set of test instances based on real world assumptions. The Austrian Red Cross provided us with data containing information about daily operations within the period of one year in the Austrian city Graz. Out of this real world data, a set of distribution parameters was extracted. The details of this data analysis process performed by Kritzinger [30] in her master thesis are summarized in Section 4.1. Using the obtained information, we created our sets of test instances as described in Section 4.2.

4.1. Data generation

As mentioned, the ARC provided us with a log file of the daily operations performed during the year 2004 in the Austrian city Graz. This sample contained information about a total of 125,035 anonymized transportation requests. The remainder of this section is a brief summary of the work performed by Kritzinger.

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 1 h. For each segment, the number of interarrival times that fall into a specific interval (e.g., 0–10 min, 10–20 min, etc.) was counted. The analysis indicated an exponential distribution. This assumption was tested using χ^2 tests. The null hypothesis was that data is exponentially distributed with a continuous density $f(x)=\lambda e^{-\lambda x}$ for $\lambda>0$ and an expected value of $E(X)=1/\lambda$. 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 $f(x)=\lambda/\Gamma(\alpha)(\lambda x)^{\alpha-1}e^{-\lambda x}$, whereby $\alpha,\lambda>0$, the expected value is $E(X)=\alpha/\lambda$, and the variance is Var $(X)=\alpha/\lambda^2$. 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 min. 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 min 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 $P(X=k)=\lambda^k e^{-\lambda}/k!$ with $k\in\mathbb{N}_0$ and an expected value of $E(X)=\lambda$. 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 toward 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.

4.2. Test instances

To test a great range of possible scenarios, we varied two parameters to come up with different sets of test instances. First, we modified the total number of transportation requests that require service during one working day of 10 h. This was achieved by modifying the distribution of the interarrival times of

incoming requests. The number of requests N was altered in the range of 100–400% of the real world size in steps of 100%. Second, we altered the probability of return transports, R. Real world data suggests that 50% of all transportation requests toward a hospital cause a return transport on the same day. We modified this parameter R to 30%, 50%, and 80%, respectively. This was used to study the sensitivity of the tested methods with respect to this factor of uncertainty. Note that our test instances do not include inbound requests that are not caused by an outbound request on the same day. We are primarily interested in the effect of stochastic information on solution quality and we do not have any stochastic information about such request. This type of requests would therefore have no effect on solution quality as it would be treated the same way as static and dynamic outbound requests.

For each of the 12 possible combinations (of one setting for the instance size and one setting for the return transport probability) we used a set of 15 test instances. These instances were created by sampling the previously found distributions with modified parameters. Starting at time $t=0$, the arrival time a_1 of the first request was determined according to the corresponding distribution of the interarrival time. Iteratively, the occurrence time of the follow-up request a_i was determined based on the arrival time of the previous request a_{i-1}. We stopped when a_i exceeded the considered period of 10 h. For each of these transportation requests, the end of the delivery time window was determined by again sampling the corresponding distribution. The length of the delivery time windows was set to 30 min. The end of the 30 min pickup time window was determined as the starting time of the delivery time window minus the time required to travel directly from pickup to delivery. The same procedure was used to generate the static transportation requests. The only difference was that, after creation, a_i was set to a value of 0 for each of the static requests.

Based on the resulting set of static and dynamic transportation requests, a set of return transports was generated. For each request, no matter whether static or dynamic, a return transport was generated with probability R. In this case, the arrival time a_r was sampled using the distribution for the time between the request's latest arrival time at the hospital and the beginning of the return transport's pickup time window. The time window for pickup at the hospital starts at a_r and was set to 60 min. The start of the time window for arrival at the patient's home location was calculated using the time required to travel directly from the hospital to the patient's home location. This time window has a length of 90 min, which is 60 min plus the maximum ride time allowed for the patient (30 min). Table 1 shows the average number of transportation requests that have to be serviced in each set of test instances (including static, dynamic, and return transports).

Finally, we created a greedy solution for each of the test instances using the modified cheapest insertion procedure described in Section 3.2 under the assumption that all requests are known a priori. The number of vehicles used in this solution was increased by 10% and used as the maximum number of vehicles for all other algorithms.

Table 1
Average number of transportation requests per instance class (R—return transport probability, N—relative size of the instance set).

N (%)	R		
	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 2
Average number of requests per type depending on return transport probability (R—return transport probability).

R (%)	Average number of requests			
	Static	Dynamic	Return	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

4.3. Parameter settings

Recent publications supply many well tested state-of-the-art solution approaches for different variants of vehicle routing problems and especially the DARP. We decided to re-use the basic design decisions and parameter settings presented by Parragh et al. [4], Gutjahr et al. [23], and Bent and Van Hentenryck [25]. Our main focus was to determine factors that influence the effect of using stochastic information while planning. Also, we wanted to present a guideline for the successful exploitation of stochastic information in the context of the DSDARP.

Still, some parameters needed to be set for our solution approaches. First, we analyzed the threshold level τ which defines when to open a new route when inserting requests into a solution. Extensive testing showed that values in the range between 20 and 30 min increase in tardiness lead to the best solution quality. As any influence on solution quality caused by varying this parameter would affect all our solution approaches equally, we decided to fix it to a value of $\tau=25$. Second, we set the maximum intensity level used for our shaking operators to $\kappa_{max}=5$.

Finally, we analyzed which influence the parameter S_{max} has on solution quality. Here, S_{max} represents the period of time for which possible scenarios of the future are taken into account. This indeed was a crucial factor for the success of using stochastic information in the planning process. Therefore, we determined if there exist preferable values for this factor. To find out which values are better than others, we fixed S_{max} to a certain value and evaluated the resulting solution quality for all 12 sets of test instances. Comparing the average solution quality achieved with different settings for S_{max} over all 180 test instances, we were able to determine if some values are better than others. A comparison of these results is given in the next section.

We also performed extensive tests using a speed up factor of 10 for the simulation time. This means that simulating one instance of 10 h is run within 1 h of CPU time. We found out that results using such a speed up factor are not representative for the results obtained when using real time. In fact, the benefit of additional runtime is much larger for dynamic S-VNS and dynamic VNS than for MPA and MSA. We found that these results would lead to completely different conclusions and thus should not be used for parameter tuning and similar activities.

5. Results

Computational testing was performed using non-parallel C++ implementations of the described algorithms. Compilation was done using the GNU C++ compiler in its version 4.1.2 on CentOS 5.5. All calculations were performed using double precision with

no rounding using one core of a SUN Fire X2270 server with 2 quad-core Intel Xeon X5550 processors (2.66 GHz) and 24 GB of shared memory.

5.1. Look ahead period

As explained in Section 4.3, the effect of taking stochastic information into account while planning the vehicle routes strongly depends on the selected value for S_{max}. We tested values of 3, 5, 10 and 20 min. As it turned out, lower values tend to lead to a better solution quality than high values. A summary of the solution quality achieved with each setting on average over all 12 sets of test instances is given in Table 3. The table reports the relative gaps in solution quality compared to 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. Note that we used a lexicographic objective function so that essentially only the first block referring to the primary objective (tardiness, T) is relevant. The blocks referring to the other objectives (number of vehicles, V, and total route duration, D) are reported for the sake of completeness. A graphical representation of these results is given in Fig. 1, in which a peak at the value of 3 min can be observed for S-VNS whereas for MSA slightly larger values for S_{max} seem to be more beneficial although MSA does not appear to be as sensitive to changes in S_{max} as dynamic S-VNS. From this we conclude that incorporating only the very near future in the planning process is most beneficial to solution quality. This might be caused by the fact that a sample based estimation of a solution's quality will most likely change very quickly as new requests arise. A summary of all solutions obtained using $S_{max}=3$ and 20 is given in Tables 4 and 5, respectively.

Table 3
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 (T—tardiness, V—number of vehicles, D—total route duration).

S_{max}	$gap(T)$ in %		$gap(V)$ in %		$gap(D)$ in %	
	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

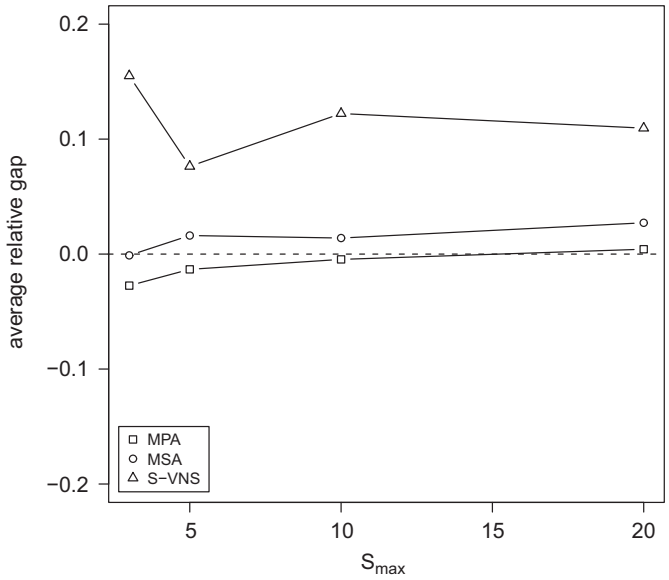

Fig. 1. Average relative gaps depending on S_{max}.

The results presented in Table 3 show that, on average over all our test instances, dynamic S-VNS can improve solution quality by 15.50% when using a sampling period of $S_{max}=3$ min. Also, MSA can improve the average solution quality by 2.72% when using $S_{max}=20$ min. This means that we were able to verify the hypothesis that taking into account stochastic information about future return transports while planning vehicle routes for the Austrian Red Cross is beneficial to solution quality. Also, we observed the design of the used methods does have a strong impact on the obtained advantages of using stochastic information. Knowing that we wanted to determine additional factors that influence the amount of improvement that can be achieved. Therefore, we needed 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 MSA and $S_{max}=3$ for S-VNS. Note that MPA, which is not influenced by S_{max}, achieved an average gap in tardiness of 0.08% over all runs included in Table 3. This indicates that the used long term memory concept does not lead to significant improvements over dynamic VNS.

5.2. Discussion of findings

Average results depending on relative test instance size N are reported in Table 7. These results show that the relative

Table 4
Summary of results obtained with $S_{max}=3$ (R—return transport probability, N—relative instance size, T—tardiness, V—number of vehicles, D—total route duration).

R	N	$gap(T)$ in %			$gap(V)$ in %			$gap(D)$ in %		
		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 5
Summary of results obtained with $S_{max} = 20$ (R—return transport probability, N—relative instance size, T—tardiness, V—number of vehicles, D—total route duration).

R	N	$gap(T)$ in %			$gap(V)$ in %			$gap(D)$ in %		
		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 6
Summary of results obtained with $S_{max} = 3$ in the unbiased test case (R—return transport probability, N—relative instance size, T—tardiness, V—number of vehicles, D—total route duration).

R	N	$gap(T)$ in %		$gap(V)$ in %		$gap(D)$ in %	
		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

advantage of dynamic S-VNS over dynamic VNS is slightly lower for larger instances. More precisely, dynamic S-VNS achieves average improvements of 11.12–25.86%. For our instance sets, MSA provides average improvements in the range from −1.35% to 9.98%. Note that we also tested the originally proposed design of S-VNS which uses more than one sample for comparing two solutions and additionally stores a best so far solution that is compared to the current solution using a different set of samples with an increasing number of elements. This version yields average improvements of 4.72–20.61% when compared to dynamic VNS and thus performs significantly worse than our modified dynamic S-VNS approach which yields average improvements of 11.12–25.86%.

An important finding is the effect that return transport probability R has on the solution quality obtained by dynamic S-VNS. This effect can be observed when looking at the average results over all test instances using the same value for R reported in Table 8. The effect can also be seen in Fig. 3. It shows the average relative gaps for dynamic S-VNS with respect to S_{max} and R. If return transports are less likely to happen, dynamic S-VNS performs clearly better than in situations with a higher probability for return transports. MSA, however, does not seem to be affected as strongly by changes in R (see Fig. 2). At first glance one would expect the opposite result: If return transports are more likely to occur, forecasts of future requests should be more important and should lead to better results. However, with a higher probability for return transports (R=80%), the number of return transports (and therefore the total number of dynamic requests) is relatively large compared to the instance size. This can also be seen in Table 2. Since a larger number of return transports implies a higher total degree of dynamism, the myopic approach (dynamic VNS) seems to perform better and the advantage of dynamic S-VNS gets smaller as the situation changes too frequently.

Yet another observation can be seen in Table 7. Dynamic S-VNS leads to solutions that on average use 7.45% more vehicles and have 35.42% longer total route durations compared to the solutions found by dynamic VNS. 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 also for most other scenarios as well. This tradeoff 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 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 MSA.

5.3. 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 [25]. The consensus over several biased solutions might strengthen the bias even more, thus leading to lower solution quality.[1]

To verify if this effect might be the reason for the observed deterioration of solution quality of dynamic S-VNS in cases with a

[1] We are grateful to one of the reviewers for pointing this issue out.

Table 7
Average solution quality depending on relative test instance size with $S_{max} = 20$ MSA and $S_{max} = 3$ for dynamic S-VNS (N—relative instance size, T—tardiness, V—number of vehicles, D—total route duration).

N(%)	$gap(T)$ in %			$gap(V)$ in %			$gap(D)$ in %			Iterations			
	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

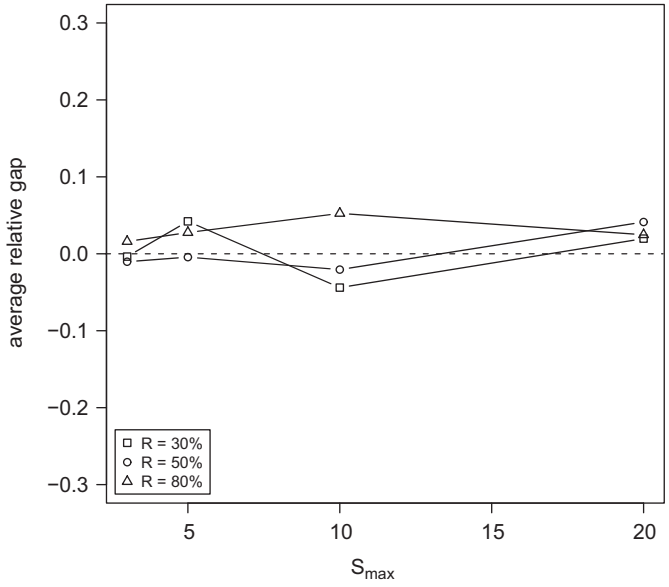

Fig. 2. Average relative gaps depending on S_{max} and R for MSA.

higher return transport probability R, we tested a variant of the stochastic algorithms that also samples 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 in the same way as the test instances were created (see Section 4.2).

The results for these unbiased tests are presented in Table 6. 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 uncertainty regarding the time of occurrence (as do the samples of inbound request) but also regarding the geographical location of upcoming requests and therefore each of the sampled requests has a very low probability to coincide with a real request. Therefore, 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 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, MSA has a better chance to correct for the bad sampling at a later point in time, which dynamic S-VNS has not.

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 MSA is the more robust method with respect to R and N, which in some cases might be favorable.

6. Summary and outlook

In this paper we compared four different metaheuristic solution approaches for the dynamic stochastic dial-a-ride problem. The main objective of our work was to verify whether taking stochastic information about future return transports into account is beneficial to solution quality. We were able to show that this was the case if certain conditions were met. 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 MPA).

We also found out that the sampling period S_{max} for possible future return transports is an important influence factor for the performance of dynamic S-VNS and MSA. More precisely, the period of time for which stochastic information about the future should be taken into account should be quite short. In our case values of up to 20 min have proven to be most effective for dynamic S-VNS and MSA, respectively. The achieved average gaps to solutions obtained by dynamic VNS reach up to 15.90% for MSA and up to 41.95% for dynamic S-VNS. However, in the worst case, average gaps can drop to −7.79% for MSA and −15.01% for dynamic S-VNS. For nearly all tested cases, dynamic S-VNS outperforms MSA and therefore seems to be the method of choice.

Another interesting finding was the fact that one of the main components proposed in the original design of MPA and MSA [25] does not seem to be as important for the DSDARP as for the VRPTW. Indeed, determining the current incumbent solution based on its objective function leads to slightly better results than using the proposed consensus function for this task. This finding may be due to the different objective functions used for the VRPTW (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 toward 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.

In future work we plan to answer the question if taking stochastic information about future stochastic time-dependent

Table 8
Average solution quality depending on return transport probability with $S_{max} = 20$ MSA and $S_{max} = 3$ for dynamic S-VNS (R—return transport probability, T—tardiness, V—number of vehicles, D—total route duration).

R(%)	$gap(T)$ in %			$gap(V)$ in %			$gap(D)$ in %			Iterations			
	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

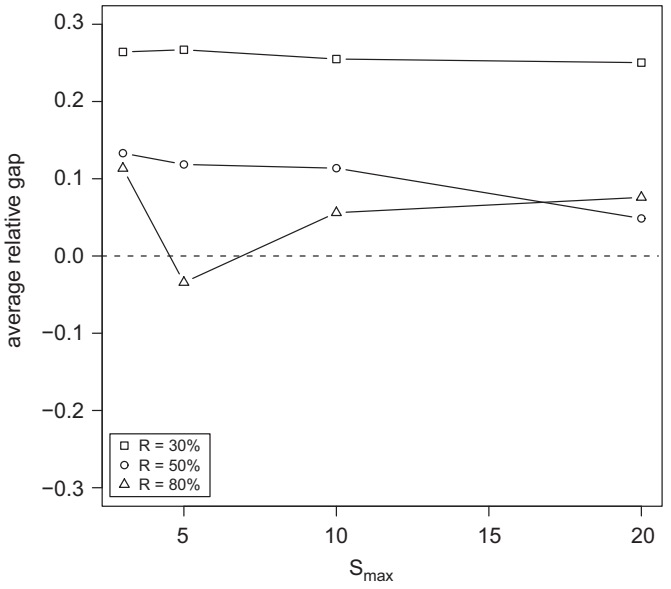

Fig. 3. Average relative gaps depending on S_{max} and R for dynamic S-VNS.

travel times can also improve solution quality and which conditions are advantageous in that case.

Acknowledgments

Financial support from the Austrian Science Fund (FWF, Translational Research) under grant #L510-N13 is gratefully acknowledged. Also, we would like to thank Stefanie Kritzinger for the data provided within the scope of her diploma thesis. The computational results presented have been achieved (in part) using the Vienna Scientific Cluster (VSC). Finally, we would like to thank two anonymous referees for their valuable input. This definitely helped to improve the quality of this paper a lot.

References

[1] Parragh S, Doerner K, Hartl R. A survey on pickup and delivery problems. Part II: transportation between pickup and delivery locations. Journal für Betriebswirtschaft 2008;58(1):81–117.
[2] Cordeau J-F, Laporte G. The dial-a-ride problem (DARP): models and algorithms. Annals of Operations Research 2007;153(1):29–46.
[3] Healy P, Moll R. A new extension of local search applied to the dial-a-ride problem. European Journal of Operational Research 1995;83(1):83–104.
[4] Parragh S, Doerner K, Hartl R. Variable neighborhood search for the dial-a-ride problem. Computers & Operations Research 2010;37(6):1129–38.
[5] Cordeau J-F, Laporte G. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transportation Research Part B: Methodological 2003;37(6):579–94.
[6] Attanasio A, Cordeau J-F, Ghiani G, Laporte G. Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. Parallel Computing 2004;30(3):377–87.
[7] Xiang Z, Chu C, Chen H. The study of a dynamic dial-a-ride problem under time-dependent and stochastic environments. European Journal of Operational Research 2008;185(2):534–51.
[8] Laporte G, Louveaux F. The integer L-shaped method for stochastic integer problems with complete recourse. Operations Research Letters 1993;13(3):133–42.
[9] Gutjahr W, Hellmayr A, Pflug G. Optimal stochastic single-machine tardiness scheduling by stochastic branch-and-bound. European Journal of Operational Research 1999;117(2):396–413.
[10] Gutjahr W. S-ACO: an ant-based approach to combinatorial optimization under uncertainty. In: 4th international workshop on ant colony optimization and swarm intelligence. Berlin, Heidelberg: Springer; 2004. p. 238–49.
[11] Jaillet P. Probabilistic traveling salesman problems. PhD thesis, MIT, Cambridge, MA; 1985.
[12] Bertsimas D. Probabilistic combinatorial optimization problems. PhD thesis, MIT, Cambridge, MA; 1988.
[13] Bertsimas D, Jaillet P, Odoni A. A priori optimization. Operations Research 1990;38(6):1019–33.
[14] Laporte G, Louveaux F, Mercure H. A priori optimization of the probabilistic traveling salesman problem. Operations Research 1994;42(3):543–9.
[15] Teodorović D, Pavković G. A simulated annealing technique approach to the vehicle routing problem in the case of stochastic demand. Transportation Planning and Technology 1992;16(4):261–73.
[16] Gendreau M, Laporte G, Séguin R. A tabu search heuristic for the vehicle routing problem with stochastic demands and customers. Operations Research 1996;44(3):469–77.
[17] Golden B, Stewart Jr W. Vehicle routing with probabilistic demands. In: Computer science and statistics: tenth annual symposium on the interface NBS special publication, vol. 503; 1978. p. 203–59.
[18] Secomandi N, Margot F. Reoptimization approaches for the vehicle-routing problem with stochastic demands. Operations Research 2009;57(1):214–30.
[19] Tillmann FA. The multiple terminal delivery problem with probabilistic demands. Transportation Science 1969;3(3):192–204.
[20] Clarke G, Wright J. Scheduling of vehicles from a central depot to a number of delivery points. Operations Research 1964;12(4):568–81.
[21] Kleywegt A, Nori V, Savelsbergh M. The stochastic inventory routing problem with direct deliveries. Transportation Science 2002;36(1):94–118.
[22] Hvattum L, Løkketangen A, Laporte G. Solving a dynamic and stochastic vehicle routing problem with a sample scenario hedging heuristic. Transportation Science 2006;40(4):421–38.
[23] Gutjahr W, Katzensteiner S, Reiter P. A VNS algorithm for noisy problems and its application to project portfolio analysis. In: 4th international symposium on stochastic algorithms: foundations and applications. Berlin, Heidelberg: Springer; 2007. p. 93–104.
[24] Mladenović N, Hansen P. Variable neighborhood search. Computers and Operations Research 1997;24(11):1097–100.
[25] Bent R, Van Hentenryck P. Scenario-based planning for partially dynamic vehicle routing with stochastic customers. Operations Research 2004;52(6):977–87.
[26] Parragh S. Introducing heterogeneous users and vehicles into models and algorithms for the dial-a-ride problem. Technical report, University of Vienna, Vienna, Austria; 2009.
[27] Parragh S, Cordeau J-F, Doerner K, Hartl R. Models and algorithms for the heterogeneous dial-a-ride problem with driver related constraints. Technical report, University of Vienna, Vienna, Austria; 2009.
[28] Rosenkrantz D, Stearns R, Lewis P. Approximate algorithms for the traveling salesperson problem. IEEE conference record of 15th annual symposium on switching and automata theory 1974:33–42.
[29] Hunsaker B, Savelsbergh M. Efficient feasibility testing for dial-a-ride problems. Operations Research Letters 2002;30(3):169–73.
[30] Kritzinger S. Warteschlangentheorie als Instrument zur Simulation von Ambulance Routing. Master's thesis, University of Salzburg, Salzburg, Austria; May 2008.