TUM School of Management

The vehicle routing problem with time windows, flexible service locations and time-dependent location capacity

Alexander Jungwirth [a,b]
Markus Frey [c]
Rainer Kolisch [a]

[a] *Technical University of Munich, Arcisstraße 21, D-80333 Munich, Germany*

[b] *BASF SE, Carl-Bosch-Straße 38, D-67056 Ludwigshafen am Rhein, Germany*

[c] *Hilti AG, Feldkircher Strasse 100, 9494 Schaan, Liechtenstein*

jungwirth.research@gmail.com
markus.frey@hilti.com
rainer.kolisch@tum.de

May 2020
TUM Technical Report (Operations Management)
OM–2020–01

Abstract: We introduce a new variant of the well-known vehicle routing problem (VRP): the VRP with time windows and flexible delivery locations (VRPTW-FL). Generally, in the VRP each customer is served in one fixed service location. However, in the VRPTW-FL each customer is served in one of a set of potential service locations, each of which has a certain capacity. From a practical point of view, the VRPTW-FL is highly relevant due to its numerous applications, e.g. parcel delivery, routing with limited parking space, and hospital-wide scheduling of physical therapists. Theoretically, the VRPTW-FL is challenging to solve due to the limited location capacities. When serving a customer, location availability must be ensured at every time. To solve this problem, we present a mathematical model and a tailored hybrid adaptive large neighborhood search. Our heuristic makes use of an innovative backtracking approach during the construction phase to alter unsatisfactory decisions at an early stage. In the meta-heuristic phase, we employ novel neighborhoods and dynamic updates of the objective violation weights. For our computational analysis, we use hospital data to evaluate the utility of flexible delivery locations and various cost functions. Our algorithmic features improve the solution quality considerably. We clearly outperform traditional hospital planning and by trading-off vehicle and customer travel times we show the economic potential of location flexibility.

Keywords: Routing, Metaheuristic, Location, OR in health services

1 Introduction

Vehicle routing is well-studied in the operations research and management science literature. It has theoretical as well as practical relevance to scientific communities and industries, such as logistics and healthcare. In the classic *vehicle routing problem* (VRP), vehicles traverse a network with the objective to e.g. minimize routing costs or the number of vehicles used. Each destination in the network corresponds to exactly one customer, and each customer is visited once. For the VRP, several extensions exist on the demand and delivery side. For example, on the delivery side assigning capacities to the vehicles leads to the *capacitated VRP*, while on the demand side associating customers with time windows leads to the *VRP with time windows* (VRPTW) (see Desaulniers et al., 2014).

In this paper, we present an extension of the VRP with substantial enhancement of the demand side: the *VRP with flexible delivery locations* (VRP-FL). In this problem, a customer is no longer automatically assigned to his/her service location. Instead, in the VRP-FL each customer must be served at exactly one *capacitated* location among a set of multiple alternatives. In this context, capacitated means that the number of customers, which can be served at one location at the same time is limited. When, additionally, time windows for customers are considered, we obtain the *VRP with time windows and flexible delivery locations* (VRPTW-FL). Note that by assigning capacities to service locations, the complexity of the problem increases significantly. Non-availability of locations leads to rerouting of customers to alternative service location. Thus, the location capacity directly influences the routing decision.

There is little literature on VRPs incorporating flexible customer locations; to the best of our knowledge, this is the first study of this type of problem with capacitated locations. The VRPTW-FL has been inspired by a problem in the health care industry, where it is known as the hospital-wide therapist scheduling and routing problem (see Gartner et al., 2018). Hospital planners have to decide which therapist treats which patient in which room at which time. Therapists can treat patients either at the ward or in a therapy center. For the VRPTW-FL, vehicles represent therapists, customers represent patients and locations for the customers represent treatments rooms. Especially in larger hospitals, the travel times of therapists are considerable. Reducing travel times allows more time to treat patients, which in the long run reduces the average waiting time for an appointment.

Another application of the VRPTW-FL is flexible parcel delivery. Companies such as DHL and Amazon have experimented with delivering to different locations depending on the time of the day (Audi AG, 2015). For example, a parcel can be sent to a customer's home, the trunk of the customer's car or to a parcel box.

This paper presents a *mixed integer program* (MIP) for the VRPTW-FL. As a generalization of the VRP, the VRPTW-FL is also $\mathcal{NP}$-hard, and as we will show, the VRPTW-FL cannot be described with a limited number of linear constraints. Both properties make this problem extremely hard to solve to optimality. Therefore, to tackle the problem we propose a *hybrid meta-heuristic* based on *adaptive large neighborhood search* (ALNS) and *guided local search* (GLS).

The construction heuristic is based on insertion, and we add a backtracking mechanism to alter unsatisfactory decisions at an early stage. The solution derived by the construction heuristic is then further improved by the hybrid ALNS. We extend the self-adaptiveness of the ALNS by allowing feasibility violations which are penalized in the objective function. The penalty weights are dynamically adjusted following a GLS approach, which adds robustness to the ALNS, and may make it more suitable for future applications.

In our computational study, we assess our algorithm from a theoretical as well as a practical perspective. In the theoretical part, we examine the performance of our heuristic procedure in general and its new features in particular. In the practical part, we test our heuristic against current hospital planning, and we evaluate the potential benefit of flexibility by applying different cost functions for serving customers in different locations and put them into relation to the vehicles' travel costs.

Our results show that the heuristic works well; combining the ALNS with a GLS leads the heuristic to considerably better regions of the planning horizon, and backtracking provides much better initial solutions than traditional construction heuristics. When applied to the hospital case, our heuristic clearly outperforms current hospital planning methods. For practitioners we provide intuition how to trade-off routing costs and customer preferences. In general, our results encourage planners facing similar problems to consider some degree of location flexibility whenever possible.

The main contribution of this paper is threefold: (a) we introduce a new variant of the VRP, which is highly relevant for practice, (b) we extend the self-adaptiveness of the ALNS by a GLS, which guides the algorithm faster to particular good regions of the solution space, and (c) we show the benefit of allowing location flexibility by employing a variety of performance metrics.

The remainder of this paper is structured as follows. We begin in §2 with an overview of related work focusing on VRPs incorporating location decisions. In §3, we develop a mathematical formulation for the VRPTW-FL and discuss the underlying graph structure. Additionally, we outline how the VRPTW-FL can be extended with multiple depots, multiple time windows, and profits. In §4, we present the hybrid meta-heuristic procedure used to solve the problem. We provide evidence of our algorithm's capabilities in §5 and conclude in §6.

2 Related work

The VRP and its extensions have been studied extensively in the literature. Textbooks include Toth and Vigo (2002, 2014) as well as Golden et al. (2008), while literature reviews are, e.g. Desrochers et al. (1990); Laporte and Osman (1995); Desrochers et al. (1999); Cordeau et al. (2002); Eksioglu et al. (2009); Laporte (2009); Lahyani et al. (2015) and Vidal et al. (2019).

Since the particular feature of the VRPTW-FL are the multiple capacitated service locations for customers, our review focuses on routing problems incorporating location decisions. The first works considering both location and routing aspects date back to the 1960s, e.g. Maranzana (1964); von Boventer (1961); Webb (1968); Watson-Gandy and Dohrn (1973). Since then a multitude of different problems have arisen, all having routing and location decisions (see e.g. Prodhon and Prins, 2014).

However, to the best of our knowledge we are the first to consider multiple capacitated locations for customers, and only two routing problems were studied in which serving *customers* is possible in multiple locations: (1) the *vehicle routing-allocation problem* (VRAP) introduced by Beasley and Nascimento (1996), and (2) the *VRP with roaming delivery locations* (VRPRDL) introduced by Reyes et al. (2017). The VRAP is a special case of the *location-routing problem* (LRP) and the VRPRDL extends the *generalized VRP* (GVRP).

In this section, we detail what, to the best of our knowledge, are the problems related to the VRPTW-FL, describe how they are connected with and how they differ from each other. Finally, we show how the VRPTW-FL generalizes all of these problems. Subsection 2.1 is devoted to LRPs and their extensions, and Subsection 2.2 focusses on GVRPs and their extensions. Figure 1 presents an overview of the evolution and the relations between the several problems that we describe in the following subsections.

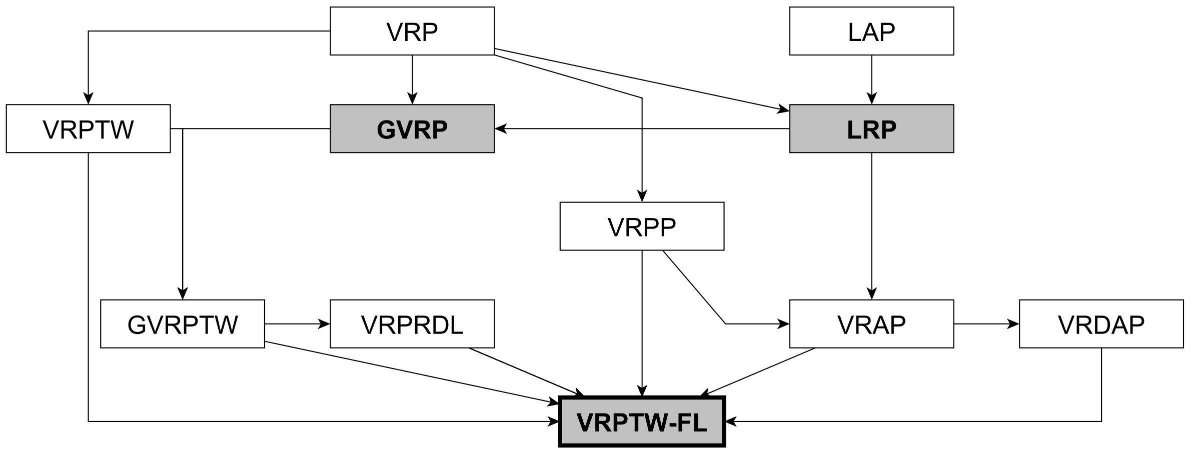


Figure 1: Connections between location and/or routing problems. (The VRPTW-FL generalizes all of them. The most relevant problems are extensions of the LRP and the GVRP (gray boxes). However, the VRPTW-FL can also be seen as an extension of the VRPTW.)

2.1 Location-routing problems

The location routing problem (LRP) can be defined as location planning incorporating tour planning (Nagy and Salhi, 2007). Generally the task is to determine locations for depots and vehicle routes from depots to customers. This problem has many practical applications, e.g. planning where distribution systems such as factories and warehouses should be placed for customers to receive their deliveries from those facilities. Reviews of LRPs are, e.g. Balakrishnan et al. (1987); Min et al. (1998); Nagy and Salhi (2007); as well as Prodhon and Prins (2014).

Clearly the LRP incorporates routing and location decisions since it combines the *location allocation problem* (LAP) and the VRP. However, the LRP and the VRPTW-FL differ considerably. While in the LRP customer locations are fixed and depot locations are flexible, the contrary is the case for the VRPTW-FL. Note that flexible depot locations could also be introduced to the VRPTW-FL as shown in Section 3.4.

The *vehicle routing-allocation problem* (VRAP) by Beasley and Nascimento (1996) is an extension of the LRP, in which customers have multiple service locations, and not all customers must be visited, i.e. a customer can be assigned to another customer's location or a customer can be left unserved. Practical applications are, e.g. routing mobile clinics in rural areas or designing postal collection routes. The location decision for customer service is very similar to the VRPTW-FL. However, there are no capacity limits for the service locations and no time windows are assigned to customers.

The VRAP is closely related to *VRPs with profits* (VRPPs), in which each customer is associated with a specific profit (see e.g. Archetti et al., 2014), and customers can be left unserved, too. The objective of the VRPP is to minimize routing costs while maximizing profits. The central difference between the VRAP and the VRPP is that the latter's customers have only one fixed service location. In the VRPTW-FL, all customers must be served. However, the functionality of having profits can easily be added (see Section 3.4).

The *vehicle routing with demand allocation problem* (VRDAP) introduced by Ghoniem et al. (2013) is a variant of the VRAP. In the VRDAP, customers are assigned to delivery sites and vehicles visit the delivery sites from a central depot. In contrast to the VRAP, the delivery sites are different from the customers' locations. One application is the distribution of food to people in need where the food is delivered e.g. to parking lots. The main difference to the VRPTW-FL is that there are no capacities in the delivery sites and no time windows for customers.

2.2 Generalized VRP

The *generalized VRP* (GVRP) constitutes the second stream of literature relevant to the VRPTW-FL. The GVRP is an extension of the VRP in which vehicles visit clusters of potential delivery sites instead of individual customers. Each cluster has a given demand and only one delivery site in the cluster must be visited; e.g. when routing vessels in maritime transportation, only one port in a certain region may have to be visited to serve the entire region. Several more practical applications exist for the GVRP (see e.g. Baldacci et al., 2010; Bektaş et al., 2011).

The GVRP can be seen as an LRP since there is a location decision to visit a particular site within a cluster. However, we believe the GVRP should be seen as an extension of the VRP as the main decision involved is the routing of vehicles, and selecting the location inside the cluster is only a minor aspect.

The GVRP is a special case of the VRPTW-FL, and the VRPTW-FL becomes a GVRP when the following two conditions are met: (1) all locations of a customer are distinct from all other locations in the problem, and (2) time windows for the customers span the entire planning horizon. Location capacity does not have to be considered since in the GVRP multiple visits to a single location are forbidden.

Moccia et al. (2012) introduce the *GVRP with time windows* (GVRPTW), where a time window is assigned to each node in the cluster and time windows of the nodes inside a cluster can differ. The VRPTW-FL cannot be transformed directly into a GVRPTW, since time windows in the VRPTW-FL are customer and not location specific, i.e. only a single time window exists for all locations of a customer. However, different time windows for different customer locations can be incorporated (see Section 3.4).

A special case of the GVRPTW that has recently attracted interest is the *VRP with roaming delivery locations* (VRPRDL) (Reyes et al., 2017; Ozbaygin et al., 2017). The problem structure is very similar to the VRPTW-FL; however, the time windows for the nodes in one cluster are disjointed. A practical application of the VRPRDL is trunk deliveries in which parcels are delivered to a customer's car which can change locations during the day. Thus multiple service locations can exist for a customer, depending on the time of day. Since the VRPRDL is a special case of the GVRPTW, transforming the VRPRDL into the VRPTW-FL is equivalent to transforming the GVRPTW into the VRPTW-FL.

3 Model development

In this section, we develop a mathematical model and discuss the underlying graph structure. Section 3.1 gives a formal problem description and introduces the notation. We follow the standard notation for VRPs and VRPTWs as presented in Irnich et al. (2014) and Desaulniers et al. (2014), respectively. However, we deviate from their notation when necessary to model the special properties of the VRPTW-FL. In Section 3.2, we detail the graph structure of the VRPTW-FL and demonstrate its differences from the graph of the classic VRPTW. Section 3.3 presents a non-linear mixed integer problem formulation for the VRPTW-FL and discusses linearizations. Finally, we show how the problem can be extended to incorporate multiple depots, multiple time windows as well as profits in Section 3.4.

3.1 Formal problem description

The classic VRP serves a set of customers $\mathcal{I} = \{1, 2 \ldots, I\}$ with specific demand $q_i > 0$ for a single good using a set of homogeneous vehicles $\mathcal{K} = \{1, \ldots, K\}$ with given capacity $Q > 0$. A vehicle starts its tour in the depot, visits a subset of customers $\mathcal{S} \subseteq \mathcal{I}$ and returns to the depot, which is denoted as dummy customer 0 for the outward trip, and $I + 1$ for the return trip. In both cases the demand is assumed to be $q_0 = q_{I+1} = 0$.

The connections between two customers i and j, including the depot as dummy customers, are associated with travel cost $c^{\text{travel}}_{l_i,l_j}$ with l_i and l_j being the service locations for customer i and j. The aggregated demand of the customers visited by a single vehicle must be less than or equal to the vehicle's capacity. The objective is to minimize the total travel costs over all vehicles while serving all customers.

The VRPTW extends the VRP by assigning a specific service time s_i and time window $[a_i, b_i]$ to each customer i, with a_i and b_i being the earliest and latest possible start of service, respectively. The travel time between two customers is denoted as $t_{i,j} \geq 0$. Generally a hard time window restriction is used, which means a vehicle can arrive at the customer before a_i but never after b_i. In case of early arrival, the vehicle must wait at the site of customer i until a_i.

The VRPTW-FL extends the VRPTW by allowing additional locations for serving the customers. The set of locations is defined as $\mathcal{L} = \{0, \ldots, L\}$ and a customer i can be served in a subset of locations $\mathcal{L}_i \subseteq \mathcal{L} \backslash \{0\}$ with location 0 being the depot. Location $l \in \mathcal{L}$ has a capacity C_l defining the maximum number of customers which can be served at the same time. For unbounded locations we set $C_l = \infty$. When serving customer i at location l, fixed location costs $c^{\text{location}}_{i,l} \geq 0$ are incurred. The location cost can be used to model that customers have preferences for certain locations. The objective of the VRPTW-FL is to minimize the sum of travel and location costs, where travel costs $c^{\text{travel}}_{l,r}$ are defined as the cost of traveling between two locations $l, r \in \mathcal{L}$ instead of two customers $i, j \in \mathcal{I} \cup \{0\}$.

The crucial information in the VRPTW-FL is if a certain location $l \in \mathcal{L}$ is available for any arbitrary small time interval $\tau \in [a_\tau, b_\tau]$ with $0 \leq a_\tau \leq b_\tau$ or if the location capacity is already fully used. Therefore, we introduce indicator function $I(i, l, k, \tau)$ which is 1 if customer i is served in location l by vehicle k in time interval τ and 0 otherwise. Using this indicator function, we can calculate the number of customers being served at a specific location in any given time interval.

Therapist scheduling as a practical application of the VRPTW-FL incorporates two additional aspects: *precedence relations* between customers and *heterogeneous vehicles*. Certain customers have to be visited before other customers can be visited. Note, in therapist scheduling a "customer" corresponds to a treatment and multiple treatments might be required for a single patient during the planning horizon. Some treatments have to be executed before other treatments can start, e.g. a cast must be removed before a stretching or strengthening exercise can be done. Therefore, we define set $\mathcal{P}$ as the precedence relations between customer tuple $\langle i, j \rangle$, in which customer i must be served before customer j can be served.

Therapists also differ in skills and shift patterns. Each therapist belongs to one of two shift types: regular shifts or short shifts. Furthermore, each therapist has a certain skill set which defines treatments that can be carried out by the therapist. Thus, it might be that a therapist is not qualified for a particular treatment or the shift pattern does not allow for visiting a customer during his/her time window. Therefore, therapists are modeled by heterogeneous vehicles and each vehicle k can service a subset of customers $\mathcal{I}_k \subseteq \mathcal{I}$.

3.2 Structural differences of VRPTW and VRPTW-FL

Having introduced the basic notation, this section examines the structural differences between the VRPTW and the VRPTW-FL. We derive a graphical representation for the VRPTW-FL and show that the optimal objective function value of the VRPTW always yields an upper bound for the VRPTW-FL.

To represent the VRPTW-FL as a network, we introduce a directed graph $G = (\mathcal{V}, \mathcal{A})$ with vertex set $\mathcal{V}$ and arc set $\mathcal{A}$. In our problem each vertex corresponds to a customer-location tuple $\langle i, l\rangle \in \{\mathcal{I} \cup \{0\}\} \times \mathcal{L}_i$. For arc set $\mathcal{A} \subseteq \mathcal{V} \times \mathcal{V}$, we have $\langle\langle i, l\rangle, \langle j, r\rangle\rangle \in \mathcal{A}$ for $\langle i, l\rangle, \langle j, r\rangle \in \mathcal{V} : i \neq j$ iff customer i can be served at location l before customer j is served at location r by the same vehicle k. Each arc is associated with a cost value $c^{\text{travel}}_{l,r}$ and a time value $t^{\text{travel}}_{l,r}$. Note that for two vertices $v_{i,l}$ and $v_{j,r}$, only locations l and r are relevant to determine the travel cost and travel time between the vertices.

Let $\mathcal{S} \subseteq \mathcal{V}$ be a subset of the vertex set. The *in-arcs*, having their head node in $\mathcal{S}$, are defined as $\delta^-(\mathcal{S}) = \{\langle v_{i,l}, v_{j,r}\rangle \in \mathcal{A} : v_{i,l} \notin \mathcal{S}, v_{j,r} \in \mathcal{S}\}$ and the *out-arcs*, having their tail node in $\mathcal{S}$, as $\delta^+(\mathcal{S}) = \{\langle v_{i,l}, v_{j,r}\rangle \in \mathcal{A} : v_{i,l} \in \mathcal{S}, v_{j,r} \notin \mathcal{S}\}$. Singleton sets $\mathcal{S} = \{v_{i,l}\}$ are defined as $\delta^{+|-}(v_{i,l}) := \delta^{+|-}(\{v_{i,l}\})$. If $\langle i, l\rangle \in \delta^-(j, r)$, then $\langle j, r\rangle \in \delta^+(i, l)$, meaning if $\langle i, l\rangle$ is a predecessor of $\langle j, r\rangle$, then $\langle j, r\rangle$ is a successor of $\langle i, l\rangle$.

If each customer can only be served at one location, then the graph of the VPRTW-FL is equal to the routing network of the VRPTW. To show the benefit of the VRPTW-FL over the VRPTW, we consider the graph in Figure 2, which shows a routing network for three customers, three service locations, and the depot. The first customer has two possible locations $\mathcal{L}_1 = \{1, 2\}$, the second customer has three possible locations $\mathcal{L}_2 = \{1, 2, 3\}$ and the third customer has one possible location $\mathcal{L}_3 = \{3\}$.[1] The time windows $[a_i, b_i]$ are given below the customer-location tuples. We assume that the travel times equal the travel costs, that the service time s_i of each customer is equal to 1, that each customer has a preferred location (marked by bold boxes), and if the customer is served in the preferred location, location costs of 0 occur and if the customer is served in another location, location costs of 1 occur. Arcs within the same location have travel costs of 0.

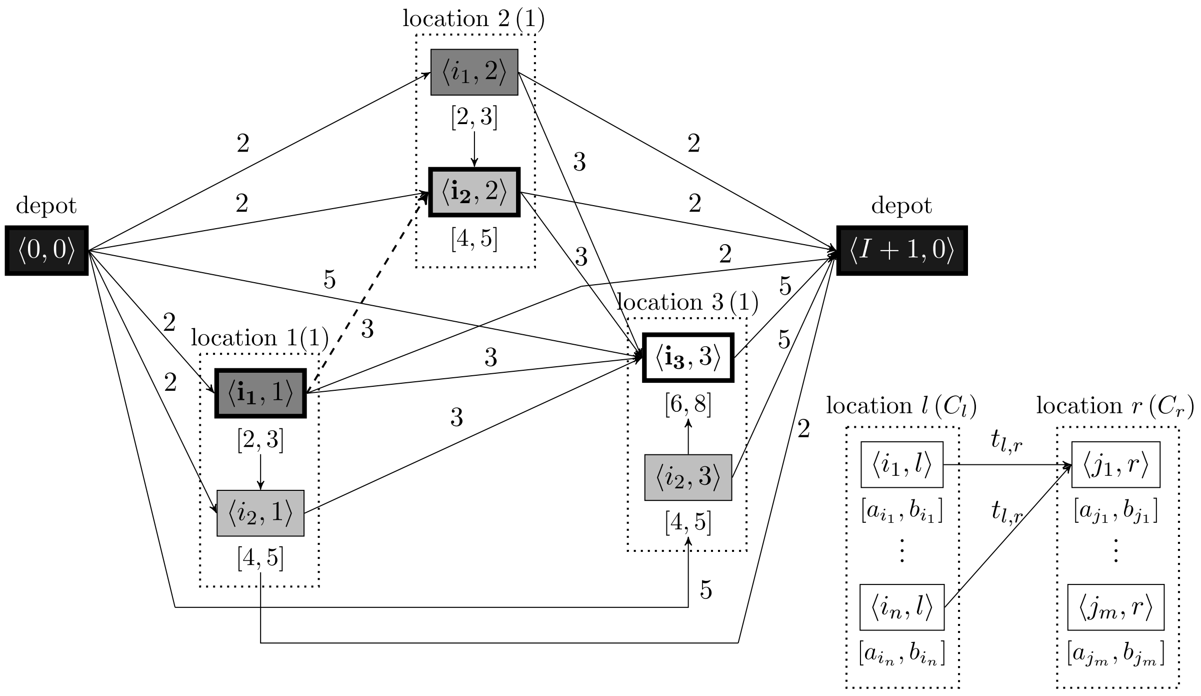


Figure 2: Routing network example: VRPTW vs. VRPTW-FL. Dotted boxes denote locations and include all customers that can be served at this location. Nodes belonging to the same customer are printed in the same color. Bold boxes denote that this location is the customer's preferred location. The dashed arrow displays the connection that is impossible due to time window restrictions. Service times are not displayed as $s_i = 1$ for all $i \in \mathcal{I}$.

[1]Note that edges between the customer-location tuples are needed to track the sequence in which customers are served. Tracking would not be possible in a network only consisting of the locations.

If, as in the VRPTW, only one location per customer exists, i.e. for the VPRTW-FL the customers must be served in their preferred location, a vehicle can either serve customers i_1 and i_3, or i_2 and i_3 but never customers i_1 and i_2. Thus, two vehicles are required leading to a total travel time of 14. In the VRPTW-FL, however, serving customer i_2 at his/her alternative location 1 guarantees that one vehicle can serve all customers within a travel time of 10 and additional location swapping cost of 1.

In general, the infeasibility of a VRPTW-FL implies the infeasibility of the corresponding VRPTW, but not vice versa. Moreover, the objective function of an optimal solution of the VRPTW yields an upper bound for the VRPTW-FL. To formalize this, we introduce function $\eta : \mathcal{I} \times \mathcal{L} \mapsto \mathcal{V}$ mapping the swap of customer i from the preferred location to another location at cost $c_{i,l}^{\text{location}}$. Then, a VRPTW-FL instance is uniquely given by tuple $\langle \mathcal{K}, \mathcal{I}, \mathcal{V}, \eta \rangle$ and Theorem 1 holds.

Theorem 1 *Having instances $\tau_1 = \langle \mathcal{K}, \mathcal{I}, \mathcal{V}, \eta \rangle$ and $\tau_2 = \langle \mathcal{K}, \mathcal{I}, \mathcal{V}', \eta \rangle$, which only differ in the vertex sets $\mathcal{V}$ and $\mathcal{V}'$ defined by the customer-location combinations, let z_1^* and z_2^* be the optimal solution for instance τ_1 and τ_2, respectively. If $\mathcal{V}_i \subseteq \mathcal{V}_i'$ holds for all $\mathcal{V}_i' \in \mathcal{V}'$ and $\mathcal{V}_i \in \mathcal{V}$ with $i \in \mathcal{I}$, then $z_2^* \geq z_1^*$.*

If each customer can be assigned to any location, i.e. all location costs $c_{i,l}^{\text{location}}$ are 0 and each location is uncapacitated, then the VRPTW-FL becomes an easy problem because all customers can be served at the location closest to the depot. However, if the location costs are greater than 0 or the locations' capacities are bounded by at least $I-1$, the VRPTP-FL is $\mathcal{NP}$-hard.

3.3 Mathematical model

For the VRPTW-FL, we have two decision variables: $x_{i,l,j,r,k} = 1$, if vehicle $k \in \mathcal{K}$ serves customer $i \in \mathcal{I}$ in location $l \in \mathcal{L}_i$ immediately before serving customer $j \in \mathcal{I}$ in location $r \in \mathcal{L}_j$, and 0 otherwise, and $T_{i,l,k}$ being the start time of serving customer $i \in \mathcal{I}$ in location $l \in \mathcal{L}_i$ by vehicle $k \in \mathcal{K}$. Binary variables $x_{i,l,j,r,k}$ constitute the tours, while continuous variables $T_{i,l,k}$ yield the scheduling decisions.

To account for heterogeneous vehicles, sets and parameters corresponding to a certain vehicle are indexed by k, e.g. $\mathcal{V}_k$ are all vertices which can be reached by vehicle k. The subset of vehicles which can serve a customer i are denoted as $\mathcal{K}_i$. Let $\mathcal{P}$ define the precedence relations between two customers $\langle i,j \rangle$. Customer j can only be served if customer i has already been served. The VRPTW-FL can now be stated as model (1)-(11):

$$\min \sum_{k \in \mathcal{K}} \sum_{\langle i,l \rangle \in \mathcal{V}_k \setminus \langle I+1,0 \rangle} \sum_{\langle j,r \rangle \in \delta_k^+(i,l)} \left(c_{l,r}^{\text{travel}} + c_{j,r}^{\text{location}} \right) \cdot x_{i,l,j,r,k} \tag{1}$$

subject to

$$\sum_{k \in \mathcal{K}_i} \sum_{\langle i,l \rangle \in \mathcal{V}_k} \sum_{\langle j,r \rangle \in \delta_k^+(i,l)} x_{i,l,j,r,k} = 1 \quad \forall\, i \in \mathcal{I} \tag{2}$$

$$\sum_{\langle j,r \rangle \in \delta_k^+(0,0)} x_{0,0,j,r,k} = 1 \quad \forall\, k \in \mathcal{K} \tag{3}$$

$$\sum_{\langle i,l \rangle \in \delta_k^-(j,r)} x_{i,l,j,r,k} - \sum_{\langle i,l \rangle \in \delta_k^+(j,r)} x_{j,r,i,l,k} = 0 \quad \forall\, k \in \mathcal{K}, \langle j,r \rangle \in \mathcal{V}_k \tag{4}$$

$$\sum_{\langle i,l \rangle \in \delta_k^-(I+1,0)} x_{i,l,I+1,0,k} = 1 \quad \forall\, k \in \mathcal{K} \tag{5}$$

$$a_i \leq T_{i,l,k} \leq b_i \quad \forall\, k \in \mathcal{K}, \langle i,l \rangle \in \mathcal{V}_k \tag{6}$$

$$x_{i,l,j,r,k} \cdot \left(T_{i,l,k} + s_i + t_{r,l}^{\text{travel}} - T_{j,r,k} \right) \leq 0 \quad \forall\, k \in \mathcal{K}, \langle i,l \rangle \in \mathcal{V}_k \setminus \langle I+1,0 \rangle, \langle j,r \rangle \in \delta_k^+(i,l) \tag{7}$$

$$T_{i,l,k_1} + s_i + t_{i,j}^{\min} \leq T_{j,r,k_2} \quad \forall\, \langle i,j \rangle \in \mathcal{P}, l \in \mathcal{L}_i, r \in \mathcal{L}_j, k_1 \in \mathcal{K}_i, k_2 \in \mathcal{K}_j \tag{8}$$

$$\sum_{i \in \mathcal{I}_l} \sum_{k \in \mathcal{K}_i} I(i,l,k,\tau) \leq C_l \quad \forall\, l \in \mathcal{L}^{\text{bounded}}, \tau \in \mathcal{T}_l^{\text{bounded}} \tag{9}$$

$$\sum_{\langle i,l\rangle \in \mathcal{V}_k} q_i \sum_{\langle j,r\rangle \in \delta_k^+(i,l)} x_{i,l,j,r,k} \leq Q_k \quad \forall\, k \in \mathcal{K} \tag{10}$$

$$x_{i,l,j,r,k} \in \{0,1\} \quad \forall\, k \in \mathcal{K}, \langle i,l\rangle \in \mathcal{V}_k, \langle j,r\rangle \in \delta_k^+(i,l) \tag{11}$$

$$T_{i,l,k} \geq 0 \quad \forall\, k \in \mathcal{K}, \langle i,l\rangle \in \mathcal{V}_k \tag{12}$$

Objective function (1) minimizes the sum of travel and location costs. The VRPTW-FL's constraint set can be divided into three parts:
Tour constraints (2)-(5): Constraints (2) ensure that every customer is served exactly once. Constraints (3)-(5) define the tour of each vehicle: constraints (3) and (5) impose a tour start and end at the depot, while constraints (4) are the flow conservation constraints.
Scheduling constraints (6)-(8): Constraints (6) set the start times for serving customer i, while constraints (7) set the time difference between two customers served consecutively by the same vehicle by linking variables $x_{i,l,j,r,k}$ and $T_{i,l,k}$. Constraints (8) ensure the precedence relations.
Location and vehicle capacity constraints (9)-(10): Constraints (9) bound the number of customers served in time interval τ at location l. Constraints (10) ensure that a vehicle k cannot satisfy more customer demand than its capacity limit Q_k. The variable domains are given in constraints (11) and (12).

Model formulation (1)-(11) is nonlinear due to constraints (7) and (9). Constraints (7) can be linearized following the approach in Desaulniers et al. (2014). However, for the linearization of constraints (9), an infinite number of linear constraints is needed, or the planning horizon must be discretized. The number of locations and customers is finite, but the number of time intervals to serve the customers is infinite due to the continuous definition of time. Therefore, if cuts were added for every location $l \in \mathcal{L}^{\text{bounded}}$, whose capacity can be violated by a subset of customers $\mathcal{S} \subseteq \mathcal{I}$ served in a specific time interval τ, an infinite number of cuts would have to be generated to enforce location capacity at each moment in time.

3.4 Generalizing the VRPTW-FL

The VRPTW-FL already adds substantial flexibility to the VRPTW. However, besides heterogeneous vehicles and precedence relations, the graph structure presented in Section 3.2 allows for further generalizations without requiring major changes to its underlying structure. We discuss three extensions: multiple depots, multiple time windows, and profits.

VRPTW-FL with multiple depots

The *multi-depot VRP* (MDVRP) has many practical applications (Renaud et al., 1996); for an overview of recent publications see Vidal et al. (2012). In the MDVRP, vehicles start their tours from more than one depot and each vehicle ends its tour in the start depot.

Multiple depots can be incorporated into the VRPTW-FL rather easily. The routing network already contains multiple service locations, and only two vertices would have to be added for each additional depot: one vertex for outbound trips and one vertex for inbound trips.

VRPFL with multiple time windows

Generally in VRPs, a customer is associated with at most one time window (Desaulniers et al., 2014). However, a few authors discuss scenarios with multiple time windows (see e.g. Ibaraki et al., 2005; Hashimoto et al., 2013). In the VRPTW-FL, all customer-location tuples have the same time window for the same customer. However, without changing the graph structure different time windows can be assigned for the different customer-location tuples of a customer. In so doing, the time window structure of the GVRPTW (Moccia et al., 2012) and the VRPRDL (Reyes et al., 2017) can be mapped.

To assign more than one time window to a customer-location tuple, simply $|\mathcal{TW}|$ copies of the customer-location tuple have to be generated where $|\mathcal{TW}|$ is the number of time windows for the specific tuple. However, the size of the graph would expand considerably.

VRPTW-FL with profits

The VRPP is a VRP in which only a subset of customers must be visited (Archetti et al., 2014). Each customer is associated with a profit, and the objective is to tradeoff the cost of traveling to the customer with the profit gained from serving the customer.

In the graph of the classic VRP, all vertices must be visited. In the VRPP, however, not all vertices must be visited. This is similar to the VRPTW-FL in which only a subset of customer-location tuples is visited. If customers were associated with profits and the constraint that all customer must be visited is relaxed, we would have a VRPTW-FL with profits without changing the structure of our solution approach, and only updated input data would be required.

4 Solution methodology

The VRPTW-FL cannot be described with a reasonable number of linear constraints (cf. §3.3), and thus generating a solution using a standard MIP solver is impossible. Therefore, our solution approach is based on an *adaptive large neighborhood search* (ALNS) framework. Using an ALNS, we can keep the continuous structure of the problem and generate a close to optimal solution relatively quickly.

The ALNS is a well-established framework for solving routing problems. It was originally developed by Ropke and Pisinger (2006a) and is still used in publications addressing new variants of VRPs (see e.g. Masson et al., 2013; Kovacs et al., 2014; Azi et al., 2014; Li et al., 2016; Mancini, 2016; Parragh and Cordeau, 2017; Schiffer and Walther, 2018). The ALNS works well for the VRPTW-FL not only due to its good performance for the VRP, but also due to the incorporation of other desirable features, such as the simplicity of the underlying concept, its flexibility with respect to VRP variants, and the possibility of using parallel hardware (Laporte et al., 2014).

The ALNS is an extension of the large neighborhood search introduced by Shaw (1998) and relies on the ruin-and-recreate principle applied in Schrimpf et al. (2000), which is similar to the rip-up principle of Dees and Karger (1982). In a first phase, an initial solution is constructed, and then in an improvement phase, the ALNS iteratively destroys parts of this solution using randomly selected destroy operators and reconstructs the destroyed solution with randomly selected repair operators. The combination of destroy and repair operators defines the neighborhood in which the new solution will be sought. If the solution is accepted according to an acceptance criterion, the current solution is replaced by the new solution and the procedure starts again. The probability of selecting a particular destroy and a repair operator is adjusted based on the success (or lack thereof) of improving a temporary solution in the past.

Our ALNS incorporates innovative features in both the construction phase and the improvement phase. In the former, our heuristic follows the k-regret insertion approach of Potvin and Rousseau (1993), which we extend with a backtracking mechanism to alter unsatisfactory decisions at an early stage. To counteract infeasible sequences of subsequently planned customers due to the simple nature of the k-regret procedure, we return to an earlier stage of the insertion with a given probability, and restart from this stage by inserting another customer.

In the improvement phase, we deviate from the standard ALNS presented by Ropke and Pisinger (2006a) in three ways: (1) we allow temporarily infeasible solutions, however sanction the infeasibilities in the objective function with penalties; (2) we dynamically adjust these penalties depending on how often certain features have been violated in past iterations; and (3) we develop new destroy operators, which exploit the underlying problem structure of potentially having more than one location per customer.

The generation of temporarily infeasible solutions enables a better traversing of the search space because it reduces the chance of getting stuck in a local optimum (Cordeau et al., 2002), and by oscillating between feasible and infeasible regions with the appropriate penalty parameters the border of feasibility is sought, a region which is very promising for finding high quality solutions (Glover and Hao, 2011; Vidal et al., 2015). For the VRPTW-FL, we allow three infeasibilities: (1) unscheduled customers, (2) violations of time windows, and (3) violations of precedence relations. In therapist scheduling, these are precisely the three aspects that a human planner would relax when faced with a hard scheduling task, where no feasible solution can be obtained manually.

Updating penalties for the violation terms dynamically extends the self-adaptiveness from operator probability updates to objective function weights, and therefore makes the approach more flexible and robust for dealing with the problem at hand. From a formal point of view, our approach combines the ALNS with a guided local search (GLS) as employed in Voudouris and Tsang (1999), and thus leads to a hybrid version of the ALNS. A simpler version of such an adaptive mechanism was originally formulated by Cordeau et al. (2001) and is e.g. used in Schiffer and Walther (2018).

In what follows, we first formalize our hybrid ALNS framework in §4.1. In §4.2, we detail the construction heuristic, including the backtracking mechanism. In §4.3, we provide the reader with information about the destroy and repair operators used, and the update procedures for the operators and objective function weights. Finally in §4.4, we provide implementation details focusing on preprocessing and parameter optimization.

4.1 Formal hybrid ALNS framework

Let s be a vector representing any (partial) solution for the VRPTW-FL and let $f(s)$ be the function that returns the objective function value for s as stated in (1), then our heuristic works on the modified objective function

$$\min f^{\text{mod}}(s) = f(s) + \lambda \cdot \sum_{i \in \mathcal{I}} \left(p_i^{\text{na}} \cdot I_i^{\text{na}}(s) + p_i^{\text{tw}} \cdot I_i^{\text{tw}}(s) + p_i^{\text{pred}} \cdot I_i^{\text{pred}}(s) \right), \tag{13}$$

where $I_i^{\text{na}}(s)$, $I_i^{\text{tw}}(s)$ and $I_i^{\text{pred}}(s)$ are indicator functions equal to 1, if in a solution s customer i is not assigned to any vehicle, if the time window of customer i is violated, and if the precedence relation of customer i is violated, respectively.[2] The penalty terms are denoted by p_i^{na}, p_i^{tw} and p_i^{pred}; see §4.3.2 for how penalties are set and updated. The weight of penalties compared to routing and location costs is controlled by λ.

Algorithm 1 provides the pseudo code for the hybrid ALNS framework. A solution is represented by s, and an initial solution s^{init} from the construction phase serves as input. The initial solution is set equal to the best global optimum found thus far (see Algorithm 1 line 1).

In the main loop 2 – 14, destroy operator $d \in \Omega^-$ and repair operators $r \in \Omega^+$ modify the current solution s^{current}. In each iteration h of the main loop, $q \geq 1$ pairs of destroy and repair operators are randomly selected to destroy and repair $n \in [\underline{n}_h, \bar{n}_h]$ elements in solution s^{current}. Parameters $\underline{n}_h$ and $\bar{n}_h$ define the lower and upper bounds of affected elements in each iteration h.

We select $q \geq 1$ different pairs of destroy and repair operators in step 3 to make use of parallel computing. The sets of destroy and repair operators are finite and will be described in detail in §4.3.3 and §4.3.4, respectively. When deriving these operators, a destroyed solution should be repairable by any repair operator. The probability of selecting a destroy operator ρ^- and a repair operator ρ^+ depends on their past success. The update procedure for the probabilities ρ^+ and ρ^- is described in §4.3.1.

If the new solution s_q^{temp} improves the best global solution s^{best}, we update s^{best} and the current solution s^{current} (see lines 6 - 7). Otherwise, we check whether the temporary solution s^{temp} is accepted as a new searching point using some criteria defined by the local search framework (see lines 8 - 10). In our case, we use a Simulated Annealing (SA) framework (see Kirkpatrick et al. (1983)), which defines the acceptance of the solution and the direction of the destroy and repair operators.

The structure of Algorithm 1 is based on Pisinger and Ropke (2010). However, the main difference is line 13, where the objective penalty terms p^{na}, p^{tw} and p^{pred} are updated (see §4.3.2 for a detailed description). The algorithm terminates after a stopping criteria has been met, e.g. a total number of iterations or iterations without improvement, then the best global solution s^{best} is returned.

Algorithm 1 Hybrid Adaptive Large Neighborhood Search

```
input: initial solution $s^{\text{init}}$ with objective $f^{\text{mod}}(s^{\text{init}})$        (see §4.2)
 1: $s^{\text{best}} = s^{\text{current}} = s^{\text{init}}$, $\rho^-(1,\dots,1)$, $\rho^+(1,\dots,1)$
 2: while stopping criteria is not met do
 3:     select $q$ pairs of destroy and repair operators $d \in \Omega^-$ and $r \in \Omega^+$ based on $\rho^-$ and $\rho^+$ (see §4.3.3
        and §4.3.4)
 4:     for each $q$ do
 5:         $s_q^{\text{temp}} = r(d(s^{\text{current}}))$
 6:         if $f^{\text{mod}}(s^{\text{best}}) < f^{\text{mod}}(s_q^{\text{temp}})$ then
 7:             $s^{\text{best}} = s^{\text{current}} = s_q^{\text{temp}}$
 8:         else if new solution $s^{\text{temp}}$ is accepted then
 9:             $s^{\text{current}} = s_q^{\text{temp}}$
10:         end if
11:         update $\rho^-$ and $\rho^+$                                        (see §4.3.1)
12:     end for
13:     update objective penalty terms $p^{\text{uns}}$, $p^{\text{tw}}$ and $p^{\text{pred}}$             (see §4.3.2)
14: end while
return: $s^{\text{best}}$
```

[2] Precedence violation of customer i is defined as service of i has started although service of preceding customer j has not been finished yet.

4.2 Construction phase

Our constructive heuristic is similar to the κ-regret[3] approach of Potvin and Rousseau (1993), which can be seen as a greedy based insertion heuristic with a look ahead perspective (see Algorithm 2). In the VRPTW-FL, the look ahead perspective becomes even more crucial than in the VRPTW as a good assignment of customers to vehicle routes may still be infeasible due to a poor assignment of customers to locations.

A route r_k for each vehicle $k \in \mathcal{K}$ is represented by an ordered sequence

$$\begin{aligned} r_k = [&\langle i_0, l_0, T_0\rangle, \dots, \langle i_{m_k-1}, l_{m_k-1}, T_{m_k-1}\rangle, \langle i_{m_k}, l_{m_k}, T_{m_k}\rangle, \\ &\langle i_{m_k+1}, l_{m_k+1}, T_{m_k+1}\rangle, \dots \langle i_{n_k}, l_{n_k}, T_{n_k}\rangle] \end{aligned}$$

of customer-location-start time tuples with $\langle i_{m_k}, l_{m_k}\rangle \in \mathcal{V}$ and $T_{m_k} \in [a_{i_m}, b_{i_m}]$. The customers in each route r_k are served according to the order given in the route sequence, i.e. for each position $0 \leq m_k \leq n_k$ in route r_k we have

$$T_{m_k-1} + s_{i_{m_k-1}} + t^{\text{travel}}_{l_{m_k-1}, l_{m_k}} \leq T_{m_k}. \tag{14}$$

At the beginning of the construction heuristic, each vehicle route r_k contains only tuples for starting and ending the tour in the depot, i.e. $\langle i_0, l_0, T_0\rangle = \langle 0, 0, 0\rangle$ and $\langle i_{n_k}, l_{n_k}, T_{n_k}\rangle = \langle n+1, 0, T\rangle$, respectively.

The goal of the heuristic is to sequentially insert one customer-location-start time tuple in one position of one of the $|\mathcal{K}|$ routes (one route for each vehicle) such that the capacities of the locations are satisfied and inequality (14) holds. However, instead of selecting the best greedy-based position within the routes, the next route position yielding the highest regret between the 1-st and the κ-th best insertion position between all routes is selected. A large gap between the best and the κ-st position indicates that a later assignment might be difficult or infeasible. The difference between our regret approach and Potvin and Rousseau (1993) is that we consider the κ best insertion positions over all routes while Potvin and Rousseau (1993) consider the κ best routes to insert a customer. Our construction heuristic works on a simplification of objective function (13) where the penalty values (costs) for each type of violation are fixed and equal for each customer.

$$\min f^{\text{simple}}(s) = f(s) + \lambda \cdot \sum_{i \in \mathcal{I}} \left(c^{\text{na}} \cdot I_i^{\text{na}}(s) + c^{\text{tw}} \cdot I_i^{\text{tw}}(s) + c^{\text{pred}} \cdot I_i^{\text{pred}}(s) \right). \tag{15}$$

Let $\mathcal{R}$ be the set of all routes over all vehicles $k \in \mathcal{K}$ and let $g_\kappa(i, \mathcal{R})$ denote the objective function value, if customer i is inserted in the κ-th best position of all routes $r_k \in \mathcal{R}$, i.e. we have $g_\kappa(i, \mathcal{R}) \leq g_{\kappa+1}(i, \mathcal{R})$ for all $r_k \in \mathcal{R}$. For objective function value $g_\kappa(i, \mathcal{R})$, we denote by $l(g_\kappa(i, \mathcal{R}))$, $T(g_\kappa(i, \mathcal{R}))$ and $m(g_\kappa(i, \mathcal{R}))$ the corresponding location, start time, and insertion position of customer i in routes $\mathcal{R}$. If only one possible insertion position is left for customer i, i.e. customer i can only be assigned and scheduled in one location and in one route at one insertion position, we set $g_\kappa(i, \mathcal{R}) = \infty$ for all $\kappa \geq 2$. Let $\mathcal{I}^{\text{na}}$ be the subset of customers, which have not yet been assigned to one route. For all routes $r_k \in \mathcal{R}$ and customer $i \in \mathcal{I}^{\text{na}}$ we compute the following regret measure:

$$\Delta g_\kappa(i, \mathcal{R}) = g_\kappa(i, \mathcal{R}) - g_1(i, \mathcal{R}). \tag{16}$$

Measure $\Delta g_\kappa(i, \mathcal{R})$ yields the difference between the best insertion position for a customer i and its κ-th best insertion position with respect to all routes. The regret for a customer indicates what can be lost in later insertions, if the customer is not immediately inserted in the best insertion position. A large regret measure indicates that the number of interesting alternative positions for inserting the customer is small, and thus this customer should be considered first. On the other hand, a small regret measure indicates that the customer can easily be inserted into alternative positions in later iterations without losing much. The customer and vehicle with the greatest regret measure is given by customer-route combination $\langle i^*, r_{k^*}\rangle = \arg\max_{i \in \mathcal{I}^{\text{na}}} \{\Delta g_\kappa(i, \mathcal{R})\}$. Thus, customer i^* is inserted in route $r_{k^*} \in \mathcal{R}$ at position $m\left(g_1(i^*, r_{k^*})\right)$; the service of customer i^* starts at time $T\left(g_1(i^*, r_{k^*})\right)$ at location $l\left(g_1(i^* r_{k^*})\right)$. If customer i cannot be inserted into any route, the regret measure $\Delta g_1(i, \mathcal{R})$ is 0 as we define that $\infty - \infty = 0$. This is either the result of a bad insertion of one or several customers in previous iterations or the instance is generally infeasible.

Let us assume that a feasible solution exists. Then, current infeasibility originates either because no insertion position exists such that inequality (14) holds or because no location is available for customer i. To provide a repair mechanism, we implement a backtracking-branching procedure.

[3]To avoid confusing indices k for vehicles and the k-regret approach, we use index κ for the k-regret approach.

Algorithm 2 illustrates the different steps of the constructive heuristic with backtracking. The initial solution s_0, only containing the depot nodes, is added to the solutions set $\mathcal{S}$, which contains all partial solutions that could not be pruned due to infeasibility (see Algorithm 2 lines 1-2). We randomly remove a customer i_1 from the set of not yet assigned customers $\mathcal{I}^{\text{na}}$, and add customer i_1 who will be served in the preferred location l_1 to route r_{k_0} (see lines 3-7). To increase diversity, and thus to find potentially better solutions, we start the entire heuristic multiple times (line 3) and perform the subsequent steps for several κ values (line 8).

While not all customers have been assigned, we calculate the regret measure for inserting every remaining customer $i \in \mathcal{I}^{\text{na}}$ in partial solution s_h (lines 9-11). If for all remaining customers a positive regret measure exists, i.e. every customer can be inserted in the partial solution s_h, the best insertion position is determined. The set of not yet assigned customers $\mathcal{I}^{\text{na}}$ and the solution set $\mathcal{S}$ are then updated (lines 12-16).

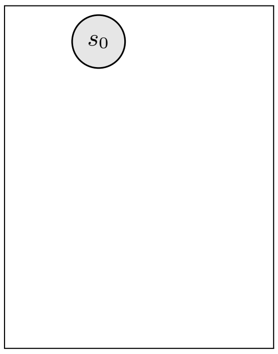
However, if for at least one customer no positive regret measure exists, i.e. this customer cannot be inserted in the partial solution, this solution becomes infeasible. We then remove s_h, the partial solution which was earlier added to the set of partial solutions $\mathcal{S}$, and return to $\text{pred}(s_h)$, the predecessor of s_h. To proceed to another solution from $\text{pred}(s_h)$, we store the deleted customer-location-start time tuple and the corresponding route r_k as tuple $\langle i_h, l_h, T_h, r_{k_h}\rangle$ in a list of forbidden insertions $\mathcal{F}(\text{pred}(s_h))$ of the partial solution $\text{pred}(s_h)$ (lines 17-20). The next insertion will then be the best customer-route combination with respect to regret measure (16) such that the corresponding customer-location start time route tuple is not contained on forbidden list $\mathcal{F}(\text{pred}(s_h))$, i.e. $(i^*, r_k^*) = \arg\max_{i\in\mathcal{I}^{\text{na}}} \{\Delta g_\kappa(i,\mathcal{R}) \mid (i, l(g_\kappa(i,\mathcal{R})), T(g_\kappa(i,\mathcal{R})), \mathcal{R}) \notin \mathcal{F}(\text{pred}(s_h))\}$.

If again no feasible successor exists, i.e. the regret measure is 0 for at least one customer, we return to $\text{pred}(s_h)$'s predecessor, for which we forbid the corresponding customer-location start time route tuple which would lead to $\text{pred}(s_h)$ again. The procedure generates a search tree in a depth-first search manner.

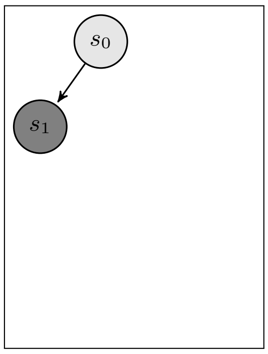
Finally, solution s^{best} having the minimal objective function value is returned (line 26). A graphic example of the backtracking mechanism is provided in Figure 3.

Algorithm 2 CONSTRUCTION PHASE

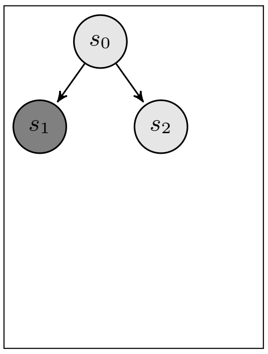
```
1: initialize partial solution $s_0$ with $r_k = [\langle 0,0,0\rangle, \langle n+1,0,T\rangle]\ \forall k \in \mathcal{K}$; $\mathcal{I}^{\text{na}} = \mathcal{I}$
2: set $\mathcal{S} = \{s_0\}$
3: while max number of restarts not reached do
4:     randomly select $i \in \mathcal{I}^{\text{na}}$; set $\mathcal{I}^{\text{na}} = \mathcal{I}^{\text{na}} \setminus \{i\}$
5:     if $\Delta g_\kappa(i,\mathcal{R}) > 0$ then
6:         $s_1 \leftarrow$ add tuple $\langle i_1, l_1(g(i,\mathcal{R}), t_1(g(i,\mathcal{R}))\rangle$ to position $m_1(g(i,\mathcal{R}))$ in route $r_{k_0} \in \mathcal{R}$ of $s_0$
7:         set $\mathcal{S} = \mathcal{S} \cup \{s_1\}$
8:         for each regret $\kappa$ do
9:             while $\mathcal{I}^{\text{na}} \neq \emptyset$ do
10:                $s_h \leftarrow$ select last inserted partial solution in $\mathcal{S}$
11:                compute regret measure for all not inserted customers
12:                if $\Delta g_\kappa(i,\mathcal{R}) > 0\ \forall i \in \mathcal{I}^{\text{na}}$ then
13:                    $(i^*, r_k^*) \leftarrow \arg\max_{i\in\mathcal{I}^{\text{na}}} \{\Delta g_\kappa(i,\mathcal{R})\}$
14:                    $s_{h+1} \leftarrow$ add tuple $\langle i^*_{h+1}, l_{h+1}(g(i^*,r_k^*)), t_{h+1}(g(i^*,r_k^*))\rangle$ to position $m_{h+1}(g(i^*,r_k^*))$ in route $r^*_{k_{h+1}}$, i.e. $r_k^* = [\langle 0,0,0\rangle, \ldots, \langle i^*, l(g(i^*,r_k^*)), t(g(i^*,r_k^*))\rangle, \ldots, \langle n+1,0,T\rangle]$
15:                    $\mathcal{I}^{\text{na}} = \mathcal{I}^{\text{na}} \setminus \{i^*\}$
16:                    set $\mathcal{S} = \mathcal{S} \cup \{s_{h+1}\}$
17:                else
18:                    set $\mathcal{S} = \mathcal{S} \setminus \{s_h\}$
19:                    return to predecessor of $\text{pred}(s_h)$
20:                    set predecessor's forbidden list $\mathcal{F}(\text{pred}(s_h)) = \mathcal{F}(\text{pred}(s_h)) \cup \{\langle i_h, l_h, T_h, r_{k_h}\rangle\}$
21:                end if
22:            end while
23:        end for
24:    end if
25: end while
26: $s^{\text{best}} \leftarrow \text{argmin}_{s\in\mathcal{S}} \{f^{\text{mod}}(s)\}$
return: $s^{\text{best}}$
```



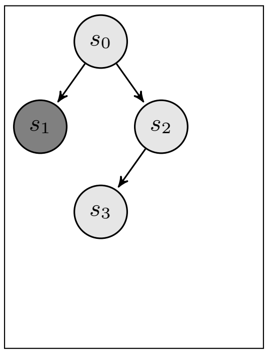
(a) Partial solution s_0



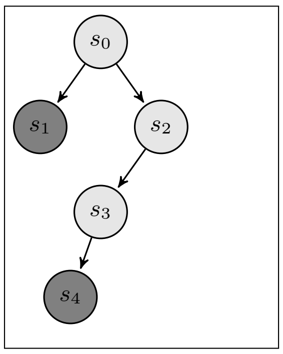
(b) Proceed to partial solution s_1, in which added tuple (i_1, l_1, T_1, r_{k_1}) leads to infeasibility



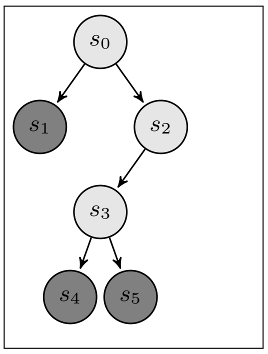
(c) Backtrack to node s_0 and set $\mathcal{F}(s_0) = \{(i_1, l_1, T_1, r_{k_1})\}$; proceed to partial solution s_2 with added tuple (i_2, l_2, T_2, r_{k_2})



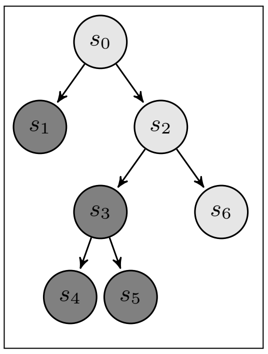
(d) Proceed to partial solution s_3 with added tuple (i_3, l_3, T_3, r_{k_3})



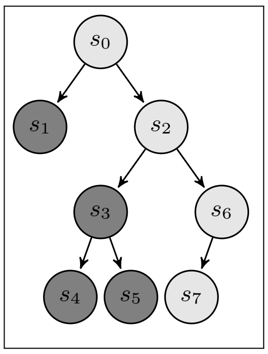
(e) Proceed to partial solution s_4, in which added tuple (i_4, l_4, T_4, r_{k_4}) leads to infeasibility



(f) Backtrack to partial solution s_2 and set $\mathcal{F}(s_3) = \{(i_4, l_4, T_4, r_{k_4})\}$; proceed to partial solution s_5, in which added tuple (i_5, l_5, T_5, r_{k_5}) leads to infeasibility



(g) Backtrack to partial solution s_3 and set $\mathcal{F}(s_3) = \mathcal{F}(s_3) \cup \{(i_5, l_5, T_5, r_{k_5})\}$; no feasible successor left; backtrack to partial solution s_2 and set $\mathcal{F}(s_2) = \{(i_3, l_3, T_3, r_{k_3})\}$; proceed to partial solution s_6 with added tuple (i_6, l_6, T_6, r_{k_6})



(h) Proceed to partial solution s_7 with added tuple (i_7, l_7, T_7, r_{k_7})

Figure 3: Example for backtracking in the constructive phase

During our first computational tests of the construction heuristic, we made the following observation: Returning to the direct predecessor of an infeasible partial solution does generally not correct the solution as desired, especially if only a few customers are left to insert. Many iterations of backtracking are needed, until a feasible solution is found. The reason is that an insertion influences the insertion position of every subsequently inserted customer. Thus, customers being inserted earlier have greater influence on the structure of the solution than customers inserted later. If poor insertion decisions have been made early, it is unlikely to correct these tens of iterations later by backtracking. Therefore, once a partial solution becomes infeasible, we do not backtrack to its immediate predecessor but to one of the first n, e.g. $n = 5$, customers inserted. By doing so, we generate high quality solutions while saving much computational time.

4.3 Operators and update functions

The algorithmic behavior of an ALNS depends heavily on (a) the destroy operators Ω^- and repair operators Ω^+ employed, i.e. the neighborhoods that can be searched, and (b) the updates of the operator weights, i.e. how fast the ALNS adjusts the probabilities of selecting a certain operator. In our hybrid ALNS, the updates of the penalty terms in the objective function also play a crucial role. In this section, we describe how our update procedures work and what operators we use. The focus lies on newly developed operators employing specific properties of the VRPTW-FL, such as multiple locations.

4.3.1 Update operator weights

To adjust the likelihood of selecting a specific operator, we follow the approach of Ropke and Pisinger (2006a). Initially all operators $j \in \Omega^{+|-}$ get assigned the same weight w_j, e.g. 1, and the probability ρ_j of selecting an operator j is:

$$\rho_j = \frac{w_j}{\sum_{i=1}^{|\Omega|} w_i} \tag{17}$$

For a given number of iterations, the success of the operators is measured by a score π_j with $j \in \Omega^{+|-}$. We distinguish four cases: (1) if a new global best solution is found, the score is raised by σ_1; (2) if a new and not yet visited solution is found with a better objective function value than the current solution, the score is raised by $\sigma_2 < \sigma_1$; (3) if a new and unvisited solution did not improve the current solution but is still accepted, the score is raised by $\sigma_3 < \sigma_2$; and (4) if a new solution is found, but this solution has already been visited in prior iterations, the score remains unchanged. Once a certain number of iterations has been reached, the operator weights are updated according to the recorded scores π_j and the counter θ_j (cf. Equation (18)). The counter θ_j measures how often the operator has been applied.

$$w_j^{\text{updated}} = w_j \cdot (1-r) + r \cdot \frac{\pi_j}{\theta_j} \tag{18}$$

Reaction factor r controls how fast the weights adapt to the success in the last iterations.

4.3.2 Update objective penalty terms

In the augmented cost function (13), penalty terms are used to penalize feasibility violations. We dynamically adjust these penalties depending on the frequency of the violation in the past and the severity of the violations. This approach of dynamically adjusting objective function weights follows the GLS employed in Voudouris and Tsang (1999) and will be described in the following.

Let f_j be a specific feature, e.g. the non-assignment of customer i_1, and indicator function $I_j(s)$ is 1, if solution s has feature f_j, and 0 otherwise. Each feature f_j, i.e. the violation of a specific constraint, is associated with a constant cost value c_j and a dynamically adjusting penalty value p_j for the objective function. All penalty values are set to 0 initially, i.e. $p_i^{\text{na}} = p_i^{\text{tw}} = p_i^{\text{pred}} = 0 \;\; \forall\, i \in \mathcal{I}$. After a certain number of iterations, the penalty values are updated for a predefined number of features yielding the highest utility value as defined in Equation (19), where s_h is the current solution at iteration h.

$$u(s_h, f_j) = I_j(s_h) \cdot \frac{c_j}{1+p_j} \tag{19}$$

The utility function is used because (a) updating all violated features equally would not change the direction of the search and lead to very similar solutions, and (b) updating only the penalties of features with the highest cost would bias the algorithm towards penalizing high cost features. The denominator $1 + p_j$ counteracts the latter since an increasing penalty p_j reduces the utility value. Note that while Voudouris and Tsang (1999) update the penalties once the heuristic is stuck in a local minumum, we update the penalties after a certain number of iterations, which is similar to updating the operator weights in an ALNS (cf. §4.3.1).

4.3.3 Destroy operators

A very useful property of the ALNS is that it can incorporate a multitude of neighborhoods to address specific characteristics of the problem at hand, and thus a multitude of destroy and repair operators have been developed (see Kovacs et al. (2014) for a good overview). In this section, we describe the destroy operators applied, and in the subsequent section the repair operators applied. As the VRPTW-FL is a generalization of the VRPTW (see §3.2), all operators are also applicable for the VRPTW. The effectiveness of the procedures will be shown in the computational study in §5.3.3. The operators used are largely taken from the literature, and adapted to the problem setting with flexible delivery locations. Furthermore, we present seven additional operators specifically designed to deal with flexible delivery locations. The operators we took from the literature and the corresponding sources are: random destroy, worst destroy (Ropke and Pisinger, 2006a), simplified Shaw (proximity) destroy, cluster destroy, history based destroy (neighbor graph destroy, request graph destroy) (Ropke and Pisinger, 2006b), related (Shaw) destroy (Shaw, 1998), and random route destroy (Mancini, 2016).

For the VRPTW-FL, the customer-locations are very important. Therefore, we introduce four operators specifically addressing the spatial arrangement of service locations: location related destroy, cluster k-means destroy, zone destroy, and subroute destroy. In addition, we use a modified time related destroy and introduce a start time flexibility destroy. For all operators incorporating some kind of relatedness, we first remove one customer-location tuple randomly and then determine the relatedness with regards to this tuple to remove further tuples.

Time related destroy In the time-related destroy operator, we select those customers i and j which have a strong relation to each other with respect to possible service times. We measure relatedness $D^{\text{time}}(i,j)$ between two customers as follows:

$$D^{\text{time}}(i,j) = \frac{T}{\left(\alpha_1 \cdot \bar{T}_{i,j} + \alpha_2 \cdot |T_i - T_j|\right)}, \tag{20}$$

where $\bar{T}_{i,j}$ is the average time difference between all possible start times of i and j:

$$\bar{T}_{i,j} = \left| \frac{a_i + b_i}{2} - \frac{a_j + b_j}{2} \right|, \tag{21}$$

and $|T_i - T_j|$ is the time difference between the start times of customers i and j in the current solution. At first, one customer i is removed at random, and then the $n-1$ customers who are most related to i are removed. This logic also applies to the other related destroy operators.

Location related destroy Similar to the time related destroy, this operator removes vertices, which are very similar in terms of their locations (cf. Equation (22)). The location relatedness $D^{\text{loc}}(i,j)$ between two customers i and j is the number of common possible service locations divided by the number of locations available for the customer with less location flexibility $(\min\{|\mathcal{L}_i|, |\mathcal{L}_j|\})$.

$$D^{\text{loc}}(i,j) = \frac{|\mathcal{L}_i \cap \mathcal{L}_j|}{\min\{|\mathcal{L}_i|, |\mathcal{L}_j|\}} \tag{22}$$

Location and time related destroy We also use the weighted combination of the location related destroy and the time related destroy:

$$D^{\text{loc,time}}(i,j) = \beta_1 \cdot D^{\text{loc}}(i,j) + \beta_2 \cdot D^{\text{time}}(i,j). \tag{23}$$

If $\beta_1 = 0$ the operator is equal to the time related destroy, if $\beta_2 = 0$ the operator is equal to the location related destroy.

Cluster destroy k-means While Ropke and Pisinger (2006b) describe a cluster destroy based on the minimum spanning tree algorithm by Kruskal (1956), we introduce a cluster destroy based on the very popular k-means clustering. The goal of k-means clustering is to partition a set into k disjoint subsets, such that the sum of the squared deviations (distances) from the positions x_j of all elements j in the clusters $\mathcal{S}_i$ to the clusters' centers μ_i is minimal. Mathematically this is:

$$\min \sum_{i=1}^{k} \sum_{j \in \mathcal{S}_i} (x_j - \mu_i)^2 . \tag{24}$$

For a recent overview of clustering algorithms and a more detailed description of k-means clustering, see Jain (2010). Depending on the underlying real-world application, it might not be possible to calculate geometric center for a subset of points. For therapist routing we use the modified Equation (25):

$$\min \sum_{i=1}^{k} \sum_{j \in \mathcal{S}_i} \left(t_{l_j, l_j^{\text{center}}}\right)^2 . \tag{25}$$

minimizing the travel time from the most centrally located location l_j^{center} to all other locations l_j in the cluster. Once the clusters have been generated, clusters are randomly selected and all customers in the selected clusters are removed until the desired number of removals has been performed. The number of cluster k can be set arbitrarily.

Zone destroy Similar to the simplified Shaw destroy, the zone destroy operator randomly selects one customer i with his/her location l_i. We then remove all customers, who *could* be assigned to one location within a given distance around location l_i. If the number of removed customers is below n, we increase the distance around l_i until n customers have been removed. Thereby, we do not only consider customers who are already close to one another but also customers who are currently served in another location but could also be served in the zone.

Subroute destroy A customer-location tuple is randomly selected and then, starting from this tuple, a virtual route of length n is constructed in a greedy fashion. Afterwards, all tuples in this virtual route are removed from the existing routes in the temporary solution.

Start time flexibility destroy In the hospital setting, customers have very different time window lengths. Outpatients generally have fixed appointments and thus a fixed start time, and some of the inpatiens have quite large time windows. The start time flexibility destroy first removes those customers who have the most flexibility in terms of possible start times. These customers are more likely to find another insertion position, while customers with fixed start times might already have a good position in the current solution.

4.3.4 Repair operators

To reinsert the removed customers, we employ two types of repair operators, namely greedy repair and κ-regret repair operators with κ ranging from 2 to 6. For the κ-regret repair, the same regret measure is used as in the construction heuristic (cf. Equation (16) in §4.2).

4.4 Implementation details

During the execution of the algorithm, feasibility must be checked frequently. Testing for capacity violations is computationally expensive; however, it does not have to be done for all customer-location combinations. Because of the start time windows and the service duration, we know for some customers that serving them in a specific location will never lead to a capacity violation, since not enough other customers exist, who could be served in this location at this specific time. Therefore, to accelerate the algorithm, we determine in a preprocessing step all locations and corresponding time intervals which could have capacity violations. During the execution of our heuristic, we only test location capacity for those customer-location combinations which could potentially lead to capacity violations.

5 Computational study

In this section, we investigate the performance of our algorithm. In particular, we describe in §5.1 how the data used in our computational experiments has been generated, and in §5.2 we detail how we adjusted the parameters of our algorithm. We evaluate our newly introduced algorithm features in §5.3. In particular, we investigate the value of (a) the backtracking procedure in the construction phase, (b) the GLS, and (c) the newly introduced destroy operators. Finally, we compare our heuristic to current hospital planning in §5.4. In addition to evaluating the solution quality, we also study the value of flexibility, i.e. how solutions change depending on different cost functions for customer travel times. Thereby, we are able to trade off customer and vehicle travel times. Finally, in §5.5, we show the performance of our algorithm for the VRPTW on the Solomon benchmark instances (Solomon, 1987).

Our algorithms were coded in JAVA using Amazon Corretto 11 as JDK and executed on a Windows 10 platform employing an Intel Core i7-4790 CPU @ 3.60GHz with 16 GB of RAM.

5.1 Data and instance generation

Since no benchmark instances exist for the VRPTW-FL, we created instances based on data provided by a cooperating hospital. To account for different problem sizes and to ensure reliability of our results, we developed an instance generator to create generic problem instances.[4] Three components define an instance: (a) the network layout representing the hospital, (b) the demand scenario representing customers (treatments), and (c) the fleet of vehicles representing therapists.

Network layout We distinguish three layouts, which are defined by the number of buildings $B \in \{1, 2, 6\}$ and the number of floors per building $F \in \{1, 3, 6\}$. Each floor has a certain number of rooms (locations) $R \in \{6, 7, \ldots, 10\}$ drawn from a discrete uniform distribution. One of the buildings contains the therapy centers, which has a capacity between 2 and 6. The capacity at the ward rooms is always unlimited since patients cannot be scheduled to other patients' ward room. The travel time between two buildings is drawn at random from the set $\{10, 15, 20\}$ minutes. The travel time between neighboring floors is assumed to be 5 minutes, and the travel time between two rooms on the same floor is either 5 or 10 minutes.

Demand scenarios We distinguish six demand scenarios having 20, 40, 60, 80, 100 and 120 treatments. A 10% probability exists that the patient is an *outpatient*, i.e. he/she can only be treated in a therapy center and the start time for the treatment is fixed. Ten percent of the patients are *bedridden*, i.e. the patient must not

[4]The problem instances are available in JSON format upon request from the corresponding author.

be moved and can only be treated in his/her room at the ward. However, these latter patients have a rather wide start time window of 90 minutes. The remaining patients are *regular inpatients*, of which 50% have location flexibility, i.e. the patient can be treated at the ward and in the therapy centers; however, a preference for one location exists, generally the ward room. The start time window length of these patients varies between 30 and 45 minutes. Thirty percent of the patients receive multiple treatments (2 or 3) in one day.

Every treatment job has a duration of 10 to 45 minutes and requires a certain skill level. We assume hierarchical skills ranging from 1 (lowest) to 3 (highest). The probabilities that a job requires a certain skill are 60%, 30% and 10% for skills 1, 2 and 3, respectively.

Vehicles We use a heterogeneous fleet, since therapists differ in their skills as well as their shift patterns. The skills are the same as for the jobs; however, the probabilities of having skill 1, 2 and 3 are 10%, 60% and 30%. A therapist has a regular (long) shift with 80% probability. Otherwise, the therapist has a short shift with 50% probability of being a morning or evening shift. We assume that therapists start and end their shifts in the break room (depot), which is 5 minutes away from the therapy centers.

Final instance set We generate two sets of data: a training set and a test set. The training set is used to pre-test the features of the heuristic and to tune its parameters, and the test set is used for the numeric study. Each set consists of $5 \cdot 3 \cdot 6 = 90$ instances, as we create five instances for each combination of the three layouts and six demand scenarios.

5.2 Hyper-parameter optimization

The performance of the ALNS strongly depends on how its parameters are adjusted, e.g. how many customers are removed in each iteration, how many parallel pairs of destroy and repair operators are evaluated, and how often the operator and penalty weights are updated. Ideally all combinations of reasonable parameter values are tested on training data, and the combination with the best performance is selected and applied to the test data. Evaluating all combinations is practically impossible. Therefore, we change only the value of one parameter in predefined steps while keeping the other parameters fixed. Once the best parameter value has been determined, the next parameter is tested, again keeping the other parameters fixed. This process could be repeated as many times as wanted. However, generally it is stopped after all parameters have been processed once (cf. Ropke and Pisinger, 2006a). We also stopped after one iteration, and our final parameters used in the numeric studies in §5.3 and §5.4 are stated in A.

5.3 Evaluation of algorithmic features

We have introduced three essential features for the ALNS: backtracking in the construction heuristic, a GLS to penalize violations of constraints, and new destroy operators specifically tailored for a problem structure, where multiple service locations exist for customers. For each feature, we first evaluate the benefit of the individual features by comparing the performance of the heuristic with and without the feature, and then we evaluate the performance of all features combined.

5.3.1 Value of backtracking

Our backtracking mechanism adds a look-ahead perspective to the construction heuristic to alter unsatisfactory decisions during the insertion process (cf. §4.2). To evaluate the value of backtracking, we compare the construction heuristic including backtracking to a version without backtracking, i.e. the best solution generated by the greedy and κ-regret insertions with $\kappa = \{2, 3, \dots, 6\}$. For backtracking, we allow stepping back to the first five inserted customers with the following probabilities 1.0, 0.6, 0.3, 0.2 and 0.1, i.e. a 10% probability exists to step back to the fifth customer, and if this is rejected, we step back to the fourth customer with a probability of 20%, etc. These probabilities are independent of one another, i.e. for each stage to which we can backtrack, a new random number is drawn if backtracking was rejected in the prior stages.

The result of the comparison of the construction heuristic with and without backtracking can be found in Table 1. For each combination of layout (B, F) and demand scenario ($|\mathcal{I}|$), we display the average values over five instances and three runs for the objective function value $f(s^*)$ of the best solution found s^*, the number of not assigned customers in that solution $|I^{\text{na}}(s^*)|$, the number of precedence violations $|I^{\text{pred}}(s^*)|$, the percentage of feasible solutions n^{feas}, and the number of backtrackings n^{bt}.

By enabling backtracking, the solution quality after the construction phase could be increased substantially. On average, the objective function value was improved by 2.4%, the numbers of not assigned customers by 70.2%,

Table 1: **Comparison of construction heuristic with and without backtracking. For each variant, the objective function value (Equation (15)), the number of not assigned customers, the number of precedence violations and the percentage of feasible solutions is displayed. For the backtracking variant, additionally the number of backtrackings is displayed. Each row represents the average values of all five instances per setting, each ran three times.**

$\lvert\mathcal{I}\rvert$	B	F	without backtracking				with backtracking				
			$f(s^*)$	$\lvert I^{\text{na}}(s^*)\rvert$	$\lvert I^{\text{pred}}(s^*)\rvert$	n^{feas}	$f(s^*)$	$\lvert I^{\text{na}}(s^*)\rvert$	$\lvert I^{\text{pred}}(s^*)\rvert$	n^{feas}	n^{bt}
20	1	6	84.67	4.80		0.73	84.47			1.00	107.80
	2	3	83.60	3.87		0.73	80.33			1.00	96.67
	6	1	81.13	3.47		0.80	76.67			1.00	110.07
40	1	6	147.47	24.20		0.33	141.93	14.27		0.60	87.80
	2	3	143.33	26.93		0.27	132.53	2.27		0.93	113.07
	6	1	141.13	25.93		0.20	125.73	11.80		0.67	78.67
60	1	6	201.87	37.73		0.27	205.73	17.53		0.67	114.20
	2	3	215.80	11.47	0.40	0.60	207.87			1.00	128.07
	6	1	215.00	44.53		0.20	201.73	21.27		0.60	94.93
80	1	6	272.33	33.33	0.13	0.47	271.13	4.33		0.93	138.40
	2	3	295.00	19.93		0.73	292.07			1.00	118.40
	6	1	267.73	23.27		0.67	263.20	12.67		0.80	102.13
100	1	6	373.60	18.47		0.80	371.47			1.00	131.93
	2	3	343.20	7.00		0.87	332.27			1.00	120.80
	6	1	338.73		0.53	0.80	326.93	14.13		0.80	87.33
120	1	6	454.67			1.00	440.27			1.00	124.87
	2	3	424.20	7.33		0.93	418.27			1.00	114.87
	6	1	417.80	37.20		0.60	419.67			1.00	150.33
Avg.			250.07	18.30	0.06	0.61	244.02	5.46	0.00	0.89	112.24
Δ							-2.4%	-70.2%	-100.0%	+45.5%	

and precedence violations were completely eliminated. The fraction of instances for which feasible solutions were found increased by 45.5% up to 89%. The number of backtrackings used per instance averaged at 112.24.

5.3.2 Value of ALNS+GLS

We tested a standard ALNS working on objective function (15) against an ALNS working on objective function (13), which dynamically adjusts penalizing infeasibilities. Since both version use different objective functions, we use the development of the best feasible solution as the metric for fair comparison. Figure 4 shows the development for all runs on a logarithmic scale. For each iteration the gap to best known solution states how far the best feasible solution of the current run is apart from the best feasible solution over all runs for the same instance. The ALNS with GLS is able to get into much better regions of the solution space and is able to improve feasible

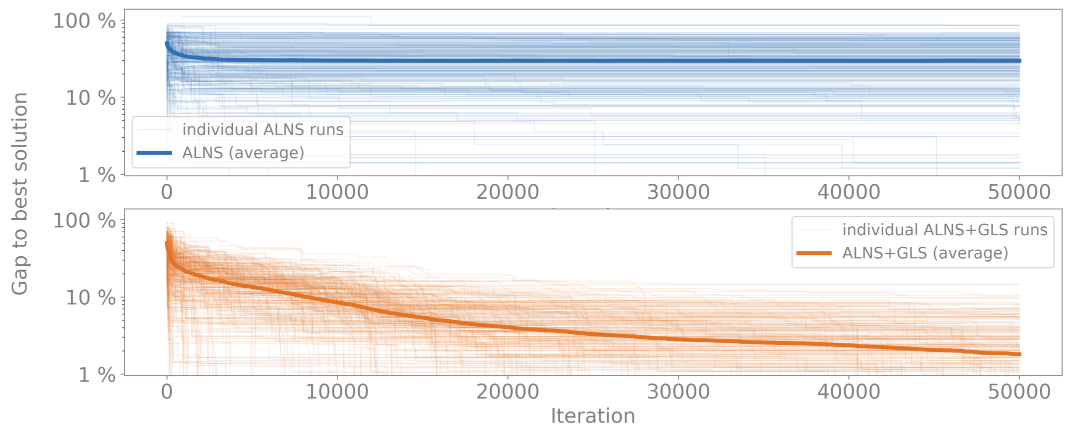


Figure 4: Comparison of ALNS (above, blue lines) and ALNS+GLS (below, orange lines). Thin lines represent the percentage gap to the best feasible solution found over all runs. The bold lines represent the average percentage gap.

solutions even in later iterations.

Table 2 shows the percentage gaps after n iterations. The gap between the two algorithms increases with increasing number of customers. The gap also increases slightly with the number of iterations, from 21.15% at $n = 10,000$ to 27.77% at $n = 50,000$.

5.3.3 Value of new operators

To specifically tackle the underlying problem structure, we have developed several new neighborhoods (cf. §4.3.3). We investigated their impact by analyzing the probability of selecting a specific operator over time. The more successful a certain operator has been in earlier iterations, the more likely it will be selected in later iterations. The results for the destroy operators are given in Figure 5 and the results for the repair operators are given in Figure 6. The graphics are both organized in the same way. Each tile highlights the selection probability for one operator averaged over all instances and all runs.

Most of the operators generally used in the literature perform well and most of our newly developed operators can add additional value. The k-means clustering is a valid alternative to clustering based on Kruskal's minimal spanning trees. Only the subroute destroy fails consistently with a probability of being used below 1%.

To our surprise, the location-related and location-and-time-related destroy operators did not perform well. A possible explanation might be, that the location that is shared most by customers is the therapy center. The therapy center, however, is capacitated which limits the amount of customers, which can be placed in this neighborhood.

With the exception of the subroute destroy, all the averaged results are close together. The main reason is that the individual performance equals out over 90 instances repeated three times. However, performance in

Table 2: Achieved gap reduction in percent after n iteration by using ALNS+GLS compared to ALNS. Each row represents the average values of all five instances per setting, each ran three times.

$\lvert\mathcal{I}\rvert$	B	F	$\Delta^{\text{gap}}_{n=10{,}000}$	$\Delta^{\text{gap}}_{n=20{,}000}$	$\Delta^{\text{gap}}_{n=30{,}000}$	$\Delta^{\text{gap}}_{n=40{,}000}$	$\Delta^{\text{gap}}_{n=50{,}000}$
20	1	6	9.96	9.87	9.97	9.59	9.98
	2	3	5.04	4.26	3.95	3.75	3.65
	6	1	6.17	6.05	5.96	5.75	5.21
40	1	6	22.21	26.70	27.34	27.59	28.23
	2	3	19.56	20.48	19.90	19.42	19.50
	6	1	8.01	9.40	9.66	9.46	9.52
60	1	6	16.44	24.18	27.85	28.25	32.12
	2	3	20.25	22.47	22.77	23.15	23.82
	6	1	14.22	22.46	23.42	25.30	26.90
80	1	6	26.46	32.11	31.78	32.42	33.09
	2	3	25.26	29.48	32.69	33.32	33.64
	6	1	21.70	25.80	27.59	27.81	28.55
100	1	6	46.44	52.12	54.76	55.34	55.97
	2	3	23.21	28.13	29.38	29.94	30.30
	6	1	20.61	24.26	24.75	25.46	25.48
120	1	6	39.35	49.02	53.06	54.67	55.52
	2	3	29.83	36.84	38.49	39.44	39.97
	6	1	26.01	34.39	36.52	37.18	38.45
Avg.			21.15	25.45	26.66	27.10	27.77

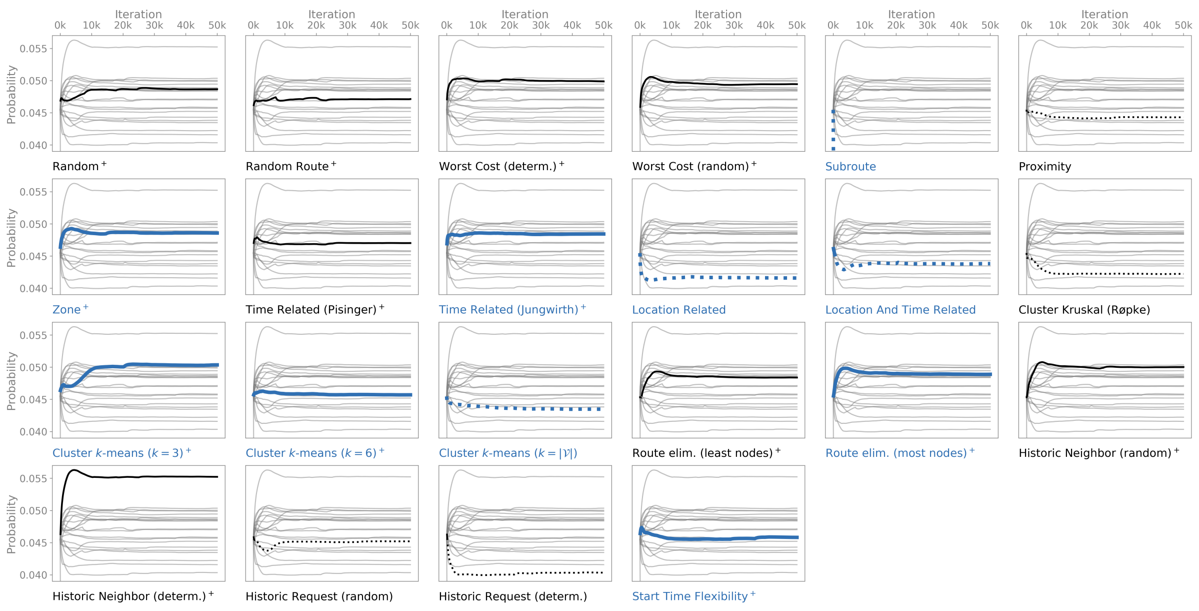


Figure 5: Probabilities of drawing the destroy operators over $50,000$ iterations. Each subplot displays the average probability of an operator over the 90 instances. Black thin lines represent operators from the literature. Blue thick lines represent our operators. Light gray lines represent the operators which are not the focus of the subplot. Above average performance is displayed by solid line segments while dotted lines represent under average performance.

the individual runs varies significantly. To present a typical example, Figure 7 displays the distribution of the individual runs for the location-related destroy.

The solution qualities of the ALNS with and without the new operators are similar. However, we observe for bigger instances (100, 120 customers) slightly better performance when using the new operators. The number of

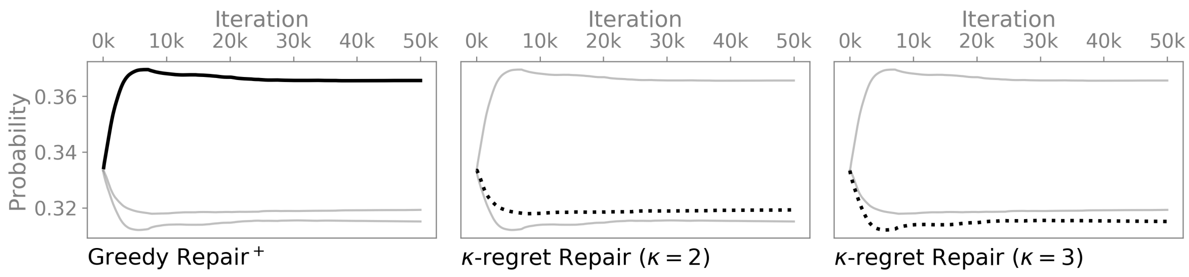


Figure 6: Probabilities of drawing the repair operators over $50,000$ iterations. Each subplot displays the average probability of an operator over the 90 instances. Black thin lines represent operators from the literature. Light gray lines represent the operators which are not the focus of the subplot. Above average performance is displayed by solid line segments while dotted lines represent under average performance.

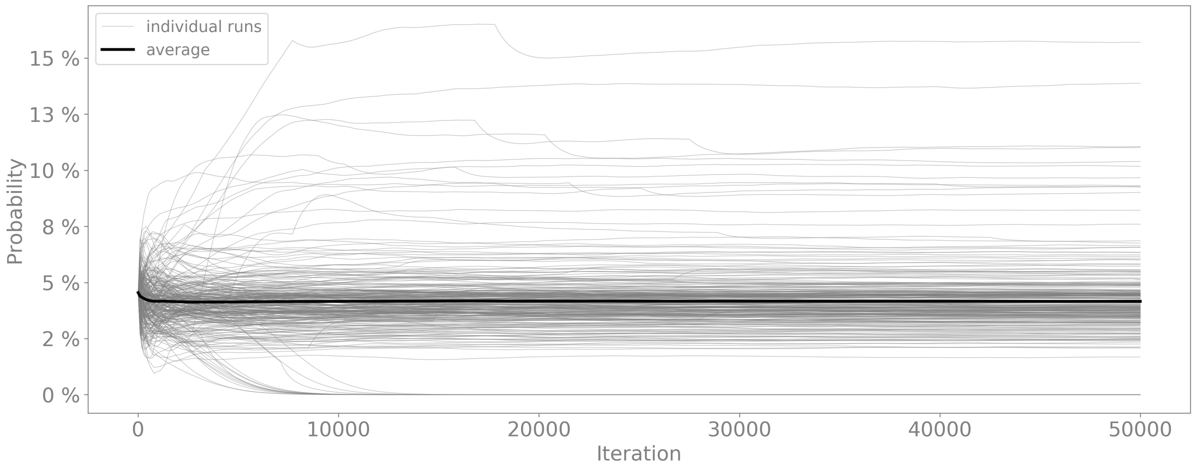


Figure 7: Distribution of the probability developments over time for the location-related destroy. Thin gray lines represent the individual runs, while the bold black line represents the average over all runs.

unscheduled customers decreases from 2.79 to 1.03 (−36.92%) and the ratio of feasible solutions found increases from 93.33% to 95.5% (+2.27%).

5.3.4 Value of all features

After having tested the algorithmic features individually, it is important to examine how the features interact when used together. The results of testing a standard ALNS against one with backtracking in the construction, GLS and new operators are displayed in Table 3.

Significant improvements are achieved when using all features. In 98% feasible solutions are generated (+4.3% over standard ALNS). The objective function value was improved by 24.9% mainly be consistently reducing the number of unscheduled customers (−99%). The number of precedence violations was also reduced by 32.5%, however in the standard ALNS it is already at a very low level of 0.02 violations per run.

5.4 VRPTW-FL applied to hospital-wide therapist scheduling and routing

After investigating the algorithmic features in the last part, this part focusses on the hospital case and the value our ALNS can add to current planning practice.

5.4.1 VRPTW-FL compared to manual planning

To compare our ALNS to current hospital planning, we use the sequential allocation heuristic (SAH) as described in Gartner et al. (2018). According to Gartner et al. (2018) this approach resembles manual planning. The central idea of the SAH is to separate customers with fixed start times, which are assigned first, from those with flexible start times, which are assigned later. Sorting is used to facilitate assignment of customers to good tours, e.g. vehicles are sorted in order of increasing shift start times and customers are sorted by earliest start time.

Table 3: Comparison of basic ALNS without new features and ALNS with all new features (i.e., backtracking in construction, additional destroy operators, and GLS). For each variant, the objective function value (Equation (15)), the number of not assigned customers, the number of time window violations, the number of precedence violations and the percentage of feasible solutions is displayed. Each row represents the average values of all five instances per setting, each ran three times.

$\|\mathcal{I}\|$	B	F	basic ALNS					ALNS with all new features				
			$f(s^*)$	$\|I^{\text{na}}(s^*)\|$	$\|I^{\text{TW}}(s^*)\|$	$\|I^{\text{pred}}(s^*)\|$	n^{feas}	$f(s^*)$	$\|I^{\text{na}}(s^*)\|$	$\|I^{\text{TW}}(s^*)\|$	$\|I^{\text{pred}}(s^*)\|$	n^{feas}
20	1	6	77.00				1.00	70.53				1.0
	2	3	68.87				1.00	66.13				1.0
	6	1	69.00				1.00	65.80				1.0
40	1	6	149.53	6.07			0.87	118.73				1.0
	2	3	127.73				1.00	107.27				1.0
	6	1	115.60	3.93			0.80	101.87				1.0
60	1	6	188.67	16.73			0.67	152.97		0.17		0.8
	2	3	178.47				1.00	140.53				1.0
	6	1	206.53				1.00	156.60				1.0
80	1	6	257.00				1.00	183.53				1.0
	2	3	254.33				1.00	195.67				1.0
	6	1	243.00	3.00			0.93	187.13				1.0
100	1	6	362.07	16.33			0.80	235.73				1.0
	2	3	282.33				1.00	214.73				1.0
	6	1	285.93			0.4	0.80	226.33	0.47		0.27	0.8
120	1	6	439.33				1.00	276.93				1.0
	2	3	355.20				1.00	268.07				1.0
	6	1	376.87				1.00	265.40				1.0
Avg.			224.30	2.56	0.00	0.02	0.94	168.55	0.03	0.01	0.02	0.98
Δ								-24.9%	-99.0%		-32.5%	+4.3%

The results for the SAH and our ALNS are given in Table 4. Our approach clearly outperforms the SAH by increasing the ratio of feasible solutions found by 29% up to 98%. Our ALNS leaves almost no customers unscheduled and reduces the number of precedence violations by 97.9%. Reducing precedence violations is highly relevant for practice because in most cases, medical reasons exist for those precedence relations.

5.4.2 Value of flexibility

Being able to assign customers to different locations provides more flexibility for the planner. Flexibility can be used to adjust the plan to the specific situation. E.g., if many outpatients are waiting for an appointment, a hospital might want to use therapists as efficient as possible, while in other cases hospitals want to avoid that patients walk unaccompanied from the ward to the therapy center and thus would be willing to accept travel times of therapists. In order to consider different situations, we evaluated four cost functions for assigning patients to a location different from the preferred location: (a) no costs, (b) small constant costs of 1 unit, (c) costs equal to the distance between preferred location and assigned location ($t_{r,l}$), and (d) costs equal to this distance squared $(t_{r,l})^2$.

Table 5 summarizes the results for the different cost functions. As expected, as changing locations becomes more expensive, more patients are treated in the preferred locations, which leads to longner travel distances for therapists. In particular, when comparing the extreme cases costs of 0 and costs of $(t_{r,l})^2$, the average ratio of patients treated in the preferred location increases by 255% from 37.8% to 96.2%, and the travel costs of the therapists increase by 224% from 71.4 to 160.04.

Figure 8 displays this causal relationship grouped by the different demand scenarios (number of customers). When costs are $(t_{r,l})^2$, deviating from the preferred locations is avoided whenever possible, only in cases of limited capacity in the therapy center an alternative location is assigned. Increasing the cost for alternative locations from 0 to 1 does only have a very small impact on the travel costs of therapists. For costs of 1, location preferences are almost entirely neglected, i.e. patients will be scheduled to alternative rooms although this only slightly improves the travel distances of the therapists. In our opinion, a good trade-off in hospital planning is to use location costs equivalent to the distance between the preferred location and the assigned locations. However, this might not generalize to other routing contexts with very different underlying network structures.

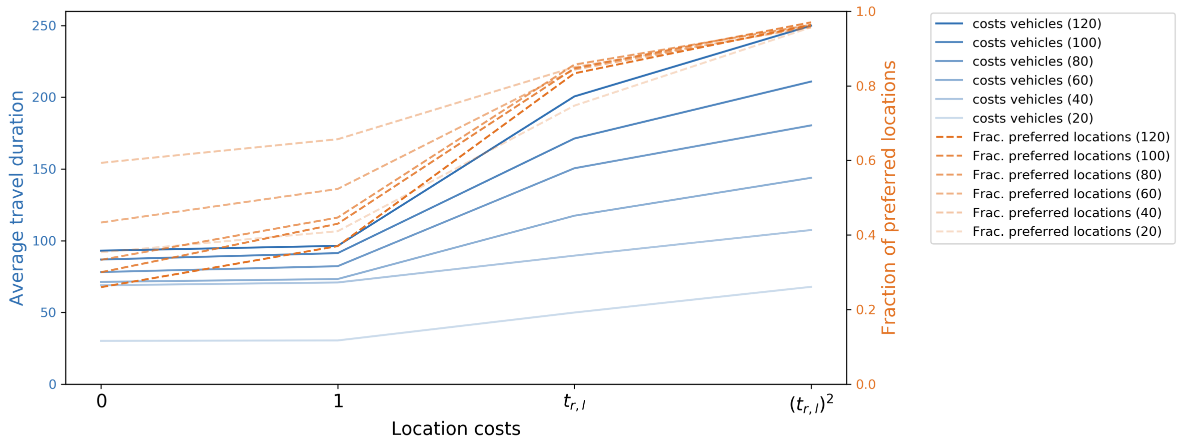


Figure 8: Travel costs of therapists (vehicles) in blue and fraction of treatments in preferred locations in orange for different location costs (0, 1, $t_{r,l}$, $(t_{r,l})^2$). Each line groups the results for different instances sizes (20, 40, $\ldots$, 120 customers).

5.5 ALNS on Solomon Instances

To show the general capabilities of our algorithm, we evaluated its performance for the VRPTW on the well-known Solomon benchmark instances (Solomon, 1987). Note that we did not tune our parameters for the Solomon instances, instead we used the parameters obtained by tuning for the hospital instances as described in §5.1. The results for the 29 instances of the first class with relatively narrow time windows is given in Table 6. Considering that our algorithm is not tuned for these instances, it performs well. For the 25-customer instances, we reach optimality in all cases, for 50 customer in 7, and for 100 customers in 2 cases. The optimality gaps are larger

Table 4: Comparison of hospital planning and ALNS. For each variant, the objective function value (Equation (15)), the number of not assigned customers, the number of time window violations, the number of precedence violations and the percentage of feasible solutions is displayed. Each row represents the average values of all five instances per setting, each ran three times.

$\lvert\mathcal{I}\rvert$	B	F	hospital planning					ALNS+GLS				
			$f(s^*)$	$I_i^{\text{na}}(s^*)$	$I_i^{\text{TW}}(s^*)$	$I_i^{\text{pred}}(s^*)$	n^{feas}	$f(s^*)$	$I_i^{\text{na}}(s^*)$	$I_i^{\text{TW}}(s^*)$	$I_i^{\text{pred}}(s^*)$	n^{feas}
20	1	6	103.2	0.2			0.8	70.53				1.0
	2	3	99.0				1.0	66.13				1.0
	6	1	92.6				1.0	65.80				1.0
40	1	6	200.6	2.2				118.73				1.0
	2	3	179.8	1.2		0.4		107.27				1.0
	6	1	175.2	0.8		0.8	0.2	101.87				1.0
60	1	6	280.4	2.0		1.2		152.97		0.17		0.8
	2	3	276.4	0.2		0.8	0.6	140.53				1.0
	6	1	278.4	2.6		0.8		156.60				1.0
80	1	6	338.0	1.2		0.4	0.2	183.53				1.0
	2	3	370.6	2.0		0.4	0.4	195.67				1.0
	6	1	355.8	2.4		0.4	0.2	187.13				1.0
100	1	6	484.2	3.0		1.6		235.73				1.0
	2	3	441.8	1.6		0.8	0.4	214.73				1.0
	6	1	429.8	1.0		2.0	0.2	226.33	0.47		0.27	0.8
120	1	6	560.2	1.8		1.2		276.93				1.0
	2	3	526.4	1.8		0.8	0.2	268.07				1.0
	6	1	521.2	3.2		1.2		265.40				1.0
Avg.			317.42	1.51	0.00	0.71	0.29	168.55	0.03	0.01	0.02	0.98
Δ								-46.9%	-98.3%		-97.9%	+238.5%

Table 5: Value of flexibility. For each setting we state the distance travelled by therapists and the fraction of treatments in preferred locations. Four different location cost functions are evaluated: no costs (0), small constant costs of 1, cost equivalent to the distance between preferred location and location of treatment $t_{r,l}$, and costs equivalent to the squared distance $(t_{r,l})^2$. Each row represents the average values of all five instances per setting, each ran three times.

$\lvert\mathcal{I}\rvert$	B	F	distance therapists				frac. preferred locations			
			0	1	$t_{r,l}$	$(t_{r,l})^2$	0	1	$t_{r,l}$	$(t_{r,l})^2$
20	1	6	28.60	28.87	55.73	65.80	0.29	0.34	0.82	0.96
	2	3	29.60	29.60	47.00	69.27	0.37	0.45	0.71	0.93
	6	1	32.33	32.87	47.00	68.40	0.40	0.44	0.71	0.98
40	1	6	70.73	74.47	95.60	113.27	0.53	0.62	0.85	0.94
	2	3	64.53	65.33	86.67	106.07	0.60	0.64	0.84	0.96
	6	1	71.40	72.80	86.47	102.93	0.65	0.71	0.86	0.97
60	1	6	75.50	77.08	123.08	139.67	0.48	0.56	0.87	0.94
	2	3	60.80	63.27	113.67	136.80	0.36	0.48	0.86	0.97
	6	1	77.53	79.47	115.47	155.00	0.46	0.53	0.80	0.97
80	1	6	73.53	78.20	149.53	175.53	0.34	0.47	0.85	0.94
	2	3	80.07	83.33	163.07	187.13	0.32	0.42	0.89	0.97
	6	1	80.80	85.13	138.87	178.47	0.34	0.45	0.83	0.98
100	1	6	92.93	97.33	193.00	226.13	0.30	0.41	0.88	0.97
	2	3	82.47	87.80	163.60	201.13	0.30	0.46	0.85	0.97
	6	1	85.17	88.83	157.08	205.50	0.30	0.42	0.81	0.97
120	1	6	101.67	106.20	219.87	258.80	0.27	0.37	0.86	0.96
	2	3	87.40	87.80	188.73	246.20	0.27	0.38	0.82	0.97
	6	1	90.07	95.20	193.07	244.67	0.24	0.36	0.82	0.96

than e.g. in Vidal et al. (2013), which is expected as our algorithm was developed to cope with the very distinct properties of the VRPTW-FL.

Table 6: Performance of our ALNS on Solomon benchmark instances. The types are clustered (C), random (R), partially random/clustered (RC), optimum is the optimal solution, ALNS is the average result achived by the ALNS over five runs, gap is the relative gap between ALNS and the optimal solution, and n optimal states how often the optimal solution was found for each instance.

type	customers	optimum	ALNS	gap	n optimal
C	25	190.59	190.59	0.0	9 / 9
	50	361.69	373.83	0.03	5 / 9
	100	826.70	944.36	0.14	1 / 9
R	25	463.37	463.37	0.0	12 / 12
	50	766.13	777.02	0.02	2 / 12
	100	1173.61	1258.68	0.08	0 / 12
RC	25	350.24	350.78	0.0	6 / 8
	50	730.31	736.14	0.01	0 / 8
	100	1334.49	1444.39	0.09	0 / 8

6 Conclusion and future work

The VRPTW-FL is a relevant and highly complex problem. Considering multiple possible service locations in routing problems receives increasing interest in the VRP community and to the best of our knowledge we are the first to address location capacities in the service locations. The VRPTW-FL occurs in scheduling physical therapists for which we have developed our solution approach. We built on an ALNS framework and enhance it with several innovative ideas, i.e. (a) a backtracking procedure in the construction phase to correct poor assignment of customers to vehicles, (b) a guided local search that dynamically adjusts how infeasibilities are penalized, and (c) new neighborhoods exploiting the underlying problem structure.

We evaluated the algorithm on a set of generic instances designed to represent different hospital layouts and different demand scenarios. Our computation results show that the developed enhancements add value to better solving the VRPTW-FL. We generated insights how different cost functions for assigning customers to location different from the preferred location affect the overall planning. These insights can be used by hospital managers to decide which cost function to be used depending on their healthcare system and/or customer preferences.

We believe that VRPTW-FL and related problems will receive more attention in near future. Savelsbergh and Woensel (2016) describe these problems as an opportunity in city logistics and we believe that for many applications location capacities become a limiting factor, e.g. limited parking space availabilities are so far completely ignored in the OR literature.

Appendix

A Overview ALNS Paramters

Table 7: ALNS parameters

$\omega = 50,000$	Number of total iterations
$\tau = 100$	Number of iterations per segment, number of iterations before probability update of operators
$r = 0.1$	Reaction parameter (roulette parameter)
$\sigma_1 = 33$	Score if new global best solution was found
$\sigma_2 = 13$	Score if new and unvisited solution was found with better objective function value than current solution
$\sigma_3 = 9$	Score if new and unvisited solution, not better than current objective value, but still accepted
$c = 0.9975$	Cooling rate
$\Delta = 0.05$	Deterioration parameters of initial solution; used to calculate start temperature
$\Omega = 0.5$	Parameter for acceptance of initial solution; used to calculate start temperature
T^{start}	Start temperature $T^{\text{start}} = -\frac{\Delta}{\ln \Omega} \cdot f(s_0)$
t^{Percent}	Auxiliary parameter to determine the end temperature
T^{end}	End temperature $T^{\text{end}} = T^{\text{start}} \cdot t^{\text{Percent}}$
$c^{\text{na}} = 3$	Costs of not assigning a customer
$c^{\text{tw}} = 0.5$	Costs of violating a time window by one time unit
$c^{\text{pred}} = 10$	Costs of violating a precedence relation
$q^{\text{lb}} = 4$	Lower bound on number of nodes that are removed from current solution
$q_1^{\text{ub}} = 0.4$	Auxiliary value to determine upper bound on number of nodes that are removed from current solution
$q_2^{\text{ub}} = 100$	Auxiliary value to determine upper bound on number of nodes that are removed from current solution
q^{ub}	Upper bound on number of nodes that are removed from current solution. $q^{\text{ub}} = \min\left\{\lvert\mathcal{I}\rvert \cdot q_1^{\text{ub}},\ q_2^{\text{ub}}\right\}$
5	Number of features for penalty update
25	Iterations between penalty updates
1	Penalty initial value
1	Penalty increase if feature is violated
0.0125	Penalty reduction if feature is not violated
5	Maximum time window violation expressed in time units
0.5	Threshold for time flexibility. Only customers i with $\frac{b_i - a_i}{T} \leq 0.5$ are allowed to have time window violations.
15	Upper bound on the number of feasible solution objects that are stored
100	Number of solutions that are considered when calculating the request graph. Needed for request graph (historic) destroy operator.
4	Zone-destroy increase factor

B Abbreviations, sets, parameters and decision variables

Table 8: Abbreviations

ALNS	Adaptive large neighborhood search
GLS	Guided local seach
GVRP	Generalized vehicle routing problem
GVRPTW	Generalized vehicle routing problem with time windows
LAP	Location allocation problem
LRP	Location routing problem
MDVRP	Multi-depot vehicle routing problem
PLRP	Periodic location routing problem
SAH	Sequential allocation heuristic
VRAP	Vehicle routing-allocation problem
VRDAP	Vehicle routing with demand allocation problem
VRP	Vehicle routing problem
VRP-FL	Vehicle routing problem with flexible delivery locations
VRPP	Vehicle routing problem with profits
VRPTW	Vehicle routing problem with time windows
VRPTW-FL	Vehicle routing problem with time windows and flexible delivery locations

Table 9: Sets, indices, parameters

δ^+	Out-arcs defined as $\delta^+(\mathcal{S}) = \{\langle v_{i,l}, v_{j,r}\rangle \in \mathcal{A} : v_{i,l} \in \mathcal{S}, v_{j,r} \notin \mathcal{S}\}$
δ^-	In-arcs defined as $\delta^-(\mathcal{S}) = \{\langle v_{i,l}, v_{j,r}\rangle \in \mathcal{A} : v_{i,l} \notin \mathcal{S}, v_{j,r} \in \mathcal{S}\}$
δ^+_k	Out-arcs for vehicle $k \in \mathcal{K}$
δ^-_k	In-arcs for vehicle $k \in \mathcal{K}$
ρ^+	Probability of a repair operator
ρ^-	Probability of a destroy operator
Ω^+	Set of repair operators
Ω^-	Set of destroy operators
$\mathcal{A}$	Set of arcs in graph G
a_i	Earliest start for serving customer $i \in \mathcal{I}$
b_i	Latest start for serving customer $i \in \mathcal{I}$
$\mathcal{C}_l$	Capacity of location $l \in \mathcal{L}$
$c^{\text{location}}_{i,l}$	Cost for serving customer $i \in \mathcal{I}$ in location $l \in \mathcal{L}$
c^{na}	Cost not assigned customer
c^{pred}	Cost for precedence violation
$c^{\text{travel}}_{l,r}$	Cost for traveling from location $l \in \mathcal{L}$ to location $r \in \mathcal{L}$
c^{tw}	Cost for time window violation
$d(\cdot)$	Destroy method (operator) $d \in \Omega^-$
$f(s)$	Objective function value of a solution s
$f^{\text{mod}}(s)$	Objective function value of a solution s for the modified objective
$f^{\text{simple}}(s)$	Simplified version of $f^{\text{mod}}(s)$ used during the construction phase and within ALNS if GLS is not used
G	A graph with $G = (\mathcal{V}, \mathcal{A})$
$\mathcal{L}$	Set of locations
$\mathcal{L}_i$	Set of potential locations for serving customer $i \in \mathcal{I}$
$\mathcal{L}^{\text{bounded}}$	Set of locations with bounded capacities
$\mathcal{I}$	Set of customers
$\mathcal{I}_k$	Set of customers that can be served by vehicle $k \in \mathcal{K}$
$\mathcal{I}_l$	Set of customers that can be served in location $l \in \mathcal{L}$
$I(\cdot)$	Indicator function
$I^{\text{na}}_i(s)$	Indicator function equal to 1, if customer i is not visited in solution s
$I^{\text{pred}}_i(s)$	Indicator function equal to 1, if precedence relation of customer i is violated in solution s
$I^{\text{tw}}_i(s)$	Indicator function equal to 1, if time window of customer i is violated in solution s
$\mathcal{K}$	Set of vehicles
$\mathcal{K}_i$	Set of vehicles which can serve customer $i \in \mathcal{I}$
$\mathcal{P}$	Set defining the precedence relations between two customers (i, j), i.e. service of customer i must be finished before service of customer j can start
p^{na}	Penalty weight not assigned customer
p^{pred}	Penalty weight for precedence violation
p^{tw}	Penalty weight for time window violation
Q_k	Capacity of vehicle $k \in \mathcal{K}$
$r(\cdot)$	repair method (operator) $r \in \Omega^+$

Sets, indices, parameters (continued)

r_k	Route of vehicle $k \in \mathcal{K}$
$\mathcal{S}$	Subset of customers with $\mathcal{S} \subseteq \mathcal{V}$
s	Some solution
s^{best}	Global best solution
s^{current}	Currently best solution
s^{init}	Initial solution
s_i	Duration for serving customer $i \in \mathcal{I}$
$t_{i,j}^{\text{min}}$	Shortest travel time from customer $i \in \mathcal{I}$ to customer $j \in \mathcal{I}$
$t_{l,r}^{\text{travel}}$	Travel time from location $l \in \mathcal{L}$ to $r \in \mathcal{L}$ location
$\mathcal{T}_l^{\text{bounded}}$	Set of continuous time intervals in which location $l \in \mathcal{L}$ has bounded capacity
$\mathcal{V}$	Set of vertices (customer-location combination)
$v_{i,l}$	Graph node representing customer-location combination (i, l) for customer $i \in \mathcal{I}$ being served in location $l \in \mathcal{L}$
$\mathcal{V}_k$	Set of vertices reachable by vehicle $k \in \mathcal{K}$

Table 10: Decision variables

$T_{i,l,k}$	Start time of serving customer $i \in \mathcal{I}$ in location $l \in \mathcal{L}$ by vehicle $k \in \mathcal{K}$
$x_{i,l,j,r,k}$	1, if vehicle $k \in \mathcal{K}$ serves customer $i \in \mathcal{I}$ in location $l \in \mathcal{L}$ right before serving customer $j \in \mathcal{I}$ in location $r \in \mathcal{L}$, 0 otherwise

References

Archetti, C., Speranza, M.G., Vigo, D., 2014. Chapter 10: Vehicle Routing Problems with Profits. Society for Industrial and Applied Mathematics, Philadelphia, PA. pp. 273–297.

Audi AG, 2015. Audi, dhl and amazon deliver convenience. URL: https://www.audi-mediacenter.com/en/press-releases/audi-dhl-and-amazon-deliver-convenience-347.

Azi, N., Gendreau, M., Potvin, J.Y., 2014. An adaptive large neighborhood search for a vehicle routing problem with multiple routes. Comput. Oper. Res. 41, 167–173.

Balakrishnan, A., Ward, J.E., Wong, R.T., 1987. Integrated facility location and vehicle routing models: Recent work and future prospects. American Journal of Mathematical and Management Sciences 7, 35–61.

Baldacci, R., Bartolini, E., Laporte, G., 2010. Some applications of the generalized vehicle routing problem. Journal of the Operational Research Society 61, 1072–1077.

Beasley, J.E., Nascimento, E.M., 1996. The vehicle routing-allocation problem: A unifying framework. TOP 4, 65–86.

Bektaş, T., Erdoğan, G., Røpke, S., 2011. Formulations and branch-and-cut algorithms for the generalized vehicle routing problem. Transportation Science 45, 299–316.

von Boventer, E., 1961. The relationship between transportation costs and location rent in transportation problems. Journal of Regional Science 3, 27–40.

Cordeau, J.F., Gendreau, M., Laporte, G., Potvin, J.Y., Semet, F., 2002. A guide to vehicle routing heuristics. Journal of the Operational Research Society 53, 512–522.

Cordeau, J.F., Laporte, G., Mercier, A., 2001. A unified tabu search heuristic for vehicle routing problems with time windows. Journal of the Operational Research Society 52, 928–936.

Dees, Jr., W.A., Karger, P.G., 1982. Automated rip-up and reroute techniques, in: Proceedings of the 19th Design Automation Conference, IEEE Press, Piscataway, NJ, USA. pp. 432–439.

Desaulniers, G., Madsen, O.B., Ropke, S., 2014. Chapter 5: The Vehicle Routing Problem with Time Windows. Society for Industrial and Applied Mathematics, Philadelphia, PA. pp. 119–159.

Desrochers, M., Jones, C., Lenstra, J., Savelsbergh, M., Stougie, L., 1999. Towards a model and algorithm management system for vehicle routing and scheduling problems. Decision Support Systems 25, 109 – 133.

Desrochers, M., Lenstra, J., Savelsbergh, M., 1990. A classification scheme for vehicle routing and scheduling problems. European Journal of Operational Research 46, 322–332.

Eksioglu, B., Vural, A.V., Reisman, A., 2009. The vehicle routing problem: A taxonomic review. Computers & Industrial Engineering 57, 1472–1483.

Gartner, D., Frey, M., Kolisch, R., 2018. Hospital-wide therapist scheduling and routing: Exact and heuristic methods. IISE Transactions on Healthcare Systems Engineering 8, 268–279.

Ghoniem, A., Scherrer, C.R., Solak, S., 2013. A specialized column generation approach for a vehicle routing problem with demand allocation. Journal of the Operational Research Society 64, 114–124.

Glover, F., Hao, J.K., 2011. The case for strategic oscillation. Annals of Operations Research 183, 163–173.

Golden, B., Raghavan, S., Wasil, E.A. (Eds.), 2008. The vehicle routing problem: Latest Advances and New Challenges. Springer US, Boston, MA.

Hashimoto, H., Yagiura, M., Imahori, S., Ibaraki, T., 2013. Recent progress of local search in handling the time window constraints of the vehicle routing problem. Annals of Operations Research 204, 171–187.

Ibaraki, T., Imahori, S., Kubo, M., Masuda, T., Uno, T., Yagiura, M., 2005. Effective local search algorithms for routing and scheduling problems with general time-window constraints. Transportation Science 39, 206–232.

Irnich, S., Toth, P., Vigo, D., 2014. Chapter 1: The Family of Vehicle Routing Problems. Society for Industrial and Applied Mathematics, Philadelphia, PA. chapter The Family of Vehicle Routing Problems. pp. 1–33.

Jain, A.K., 2010. Data clustering: 50 years beyond k-means. Pattern Recognition Letters 31, 651 – 666.

Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. Science 220, 671–680.

Kovacs, A.A., Parragh, S.N., Hartl, R.F., 2014. A template-based adaptive large neighborhood search for the consistent vehicle routing problem. Networks 63, 60–81.

Kruskal, J.B., 1956. On the shortest spanning subtree of a graph and the traveling salesman problem. Proceedings of the American Mathematical Society 7, 48–50.

Lahyani, R., Khemakhem, M., Semet, F., 2015. Rich vehicle routing problems: From a taxonomy to a definition. European Journal of Operational Research 241, 1 – 14.

Laporte, G., 2009. Fifty years of vehicle routing. Transportation Science 43, 408–416.

Laporte, G., Osman, I.H., 1995. Routing problems: A bibliography. Annals of Operations Research 61, 227–262.

Laporte, G., Ropke, S., Vidal, T., 2014. Chapter 4: Heuristics for the vehicle routing problem. Society for Industrial and Applied Mathematics, Philadelphia, PA. pp. 87–116.

Li, Y., Chen, H., Prins, C., 2016. Adaptive large neighborhood search for the pickup and delivery problem with time windows, profits, and reserved requests. European Journal of Operational Research 252, 27 – 38.

Mancini, S., 2016. A real-life multi depot multi period vehicle routing problem with a heterogeneous fleet: Formulation and adaptive large neighborhood search based matheuristic. Transportation Research Part C: Emerging Technologies 70, 100–112.

Maranzana, F.E., 1964. On the location of supply points to minimize transport costs. OR 15, 261–270.

Masson, R., é, F.L., Péton, O., 2013. An adaptive large neighborhood search for the pickup and delivery problem with transfers. Transportation Science 47, 344–355.

Min, H., Jayaraman, V., Srivastava, R., 1998. Combined location-routing problems: A synthesis and future research directions. European Journal of Operational Research 108, 1 – 15.

Moccia, L., Cordeau, J.F., Laporte, G., 2012. An incremental tabu search heuristic for the generalized vehicle routing problem with time windows. Journal of the Operational Research Society 63, 232–244.

Nagy, G., Salhi, S., 2007. Location-routing: Issues, models and methods. European Journal of Operational Research 177, 649 – 672.

Ozbaygin, G., Karasan, O.E., Savelsbergh, M., Yaman, H., 2017. A branch-and-price algorithm for the vehicle routing problem with roaming delivery locations. Transportation Research Part B: Methodological 100, 115 – 137.

Parragh, S.N., Cordeau, J.F., 2017. Branch-and-price and adaptive large neighborhood search for the truck and trailer routing problem with time windows. Computers & Operations Research 83, 28 – 44.

Pisinger, D., Ropke, S., 2010. Large Neighborhood Search. Springer US, Boston, MA. pp. 399–419.

Potvin, J.Y., Rousseau, J.M., 1993. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. European Journal of Operational Research 66, 331 – 340.

Prodhon, C., Prins, C., 2014. A survey of recent research on location-routing problems. European Journal of Operational Research 238, 1 – 17.

Renaud, J., Laporte, G., Boctor, F.F., 1996. A tabu search heuristic for the multi-depot vehicle routing problem. Computers & Operations Research 23, 229 – 235.

Reyes, D., Savelsbergh, M., Toriello, A., 2017. Vehicle routing with roaming delivery locations. Transportation Research Part C: Emerging Technologies 80, 71 – 91.

Ropke, S., Pisinger, D., 2006a. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science 40, 455–472.

Ropke, S., Pisinger, D., 2006b. A unified heuristic for a large class of vehicle routing problems with backhauls. European Journal of Operational Research 171, 750–775. Feature Cluster: Heuristic and Stochastic Methods in Optimization Feature Cluster: New Opportunities for Operations Research.

Savelsbergh, M., Woensel, T.V., 2016. 50th anniversary invited article - city logistics: Challenges and opportunities. Transportation Science 50, 579–590.

Schiffer, M., Walther, G., 2018. An adaptive large neighborhood search for the location-routing problem with intra-route facilities. Transportation Science 52, 331–352.

Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., Dueck, G., 2000. Record breaking optimization results using the ruin and recreate principle. Journal of Computational Physics 159, 139 – 171.

Shaw, P., 1998. Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 417–431.

Solomon, M.M., 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research 35, 254–265.

Toth, P., Vigo, D., 2002. The Vehicle Routing Problem. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Toth, P., Vigo, D., 2014. Vehicle Routing. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Vidal, T., Crainic, T.G., Gendreau, M., Lahrichi, N., Rei, W., 2012. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. Operations Research 60, 611–624.

Vidal, T., Crainic, T.G., Gendreau, M., Prins, C., 2013. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. Computers & Operations Research 40, 475 – 489.

Vidal, T., Crainic, T.G., Gendreau, M., Prins, C., 2015. Time-window relaxations in vehicle routing heuristics. Journal of Heuristics 21, 329–358.

Vidal, T., Laporte, G., Matl, P., 2019. A concise guide to existing and emerging vehicle routing problem variants. European Journal of Operational Research .

Voudouris, C., Tsang, E., 1999. Guided local search and its application to the traveling salesman problem. European Journal of Operational Research 113, 469 – 499.

Watson-Gandy, C., Dohrn, P., 1973. Depot location with van salesmen - a practical approach. Omega 1, 321 – 329.

Webb, M.H.J., 1968. Cost functions in the location of depots for multiple-delivery journeys. Journal of the Operational Research Society 19, 311–320.