Fakultät für Wirtschaftswissenschaften

Vehicle Routing with Time Windows and Flexible Delivery Locations

Alexander Jungwirth, Master of Science

Vollständiger Abdruck der von der Fakultät für Wirtschaftswissenschaften der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor der Wirtschaftswissenschaften (Dr. rer. pol.)

genehmigten Dissertation.

Vorsitzender	Prof. Dr. Martin Grunow
Prüfer der Dissertation	1. Prof. Dr. Rainer Kolisch
	2. Prof. Dr. Maximilian Schiffer

Die Dissertation wurde am 19.05.2020 bei der Technischen Universität München eingereicht und durch die Fakultät für Wirtschaftswissenschaften am 15.06.2020 angenommen.

Abstract

Motivated by the hospital-wide scheduling of physical therapists, we study a new variant of the well-known vehicle routing problem (VRP): the VRP with time windows and flexible delivery locations (VRPTW-FL). In the classic 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 and routing of physical therapists. Theoretically, the VRPTW-FL is challenging to solve due to the time-dependent location capacities. When serving a customer, location availability must be ensured at every time. Precedence relations between customers, flexible service locations, and a heterogeneous fleet increase the complexity further.

To solve the VRPTW-FL, we develop two mathematical models, and present a hybrid adaptive large neighborhood search and an exact branch-price-and-cut framework, Our heuristic employs an innovative backtracking procedure during the construction phase to alter unsatisfactory decisions at an early stage. In the metaheuristic phase, we employ novel neighborhoods and dynamic updates of the objective violation weights. Our exact BPC includes two innovative approaches to target the location capacity and the precedence relations: (1) based on branching on time windows, and (2) based on adding combinatorial Benders cuts.

For our computational study, we use hospital data to evaluate the benefit of flexible service locations and various cost functions. We show that our algorithmic features improve the solution quality considerably. We clearly outperform traditional hospital planning, and by trading-off vehicle travel times and preferences for service locations we show the economic potential of location flexibility. Our branch-price-and-cut framework optimally solves realistic hospital instances with up to 120 treatments, and we find that branching on time windows outperforms adding cutting planes.

Deutsch: Angelehnt an die krankenhausweite Planung von Physiotherapeuten untersuchen wir eine neue Variante des Tourenplanungsproblems (TPP): das TPP mit Zeitfenstern und flexiblen Lieferorten (TPPZF-FL). In der Grundvariante des TPP werden Kunden nur an einem Standort bedient. Im TPPZF-FL werden Kunden jedoch an einem von mehreren potentiellen Standorten bedient, von denen jeder Lieferort eine bestimmte Kapazität hat. Aus praktischer Sicht ist das TPPZF-FL relevant, da es neben der Planung von Physiotherapeuten auch weitere Anwendungen z.B. bei der Paketzustellung oder Tourenplanung mit begrenzter Parkplatzkapazität hat. Aus theoretischer Sicht ist das TPPZF-FL aufgrund der zeitabhängigen Standortkapazitäten schwer zu lösen. Um einen Kunden bedienen zu können, muss stets die Verfügbarkeit des Standorts gewährleistet sein. Vorgangsbeziehungen zwischen den Kunden, die Flexibilität der Lieferorte und eine heterogene Fahrzeugflotte erhöhen die Komplexität zusätzlich.
Um das TPPZF-FL zu lösen entwickeln wir zwei mathematische Modelle und präsentieren eine hybride Adaptive Large Neighborhood Search und ein exaktes Branch-Price-und-Cut Framework. Unsere Heuristik nutzt einen innovativen Backtracking-Ansatz während der Konstruktionsphase, um ungünstige Entscheidungen frühzeitig zu korrigieren. In der Metaheuristik-Phase verwenden wir neuartige Nachbarschaften und adjustieren dynamisch die Gewichtung von Straftermen in der Zielfunktion. Unser exaktes Branch-Price-und-Cut Framework beinhaltet zwei innovative Ansätze um die Standortkapazität und die Vorgangsbeziehungen zu berücksichtigen: (1) basierend auf dem Branchen auf Zeitfenstern und (2) basierend auf dem Hinzufügen von kombinatorischen Benders Schnittebenen.
In der Rechenstudie verwenden wir Krankenhausdaten, um den Wert von flexiblen Lieferorten und unterschiedlichen Kostenfunktionen zu quantifizieren. Wir zeigen, dass die algorithmischen Erweiterungen die Lösungsqualität erheblich verbessern. Wir übertreffen die aktuelle Krankenhausplanung deutlich und zeigen durch Abwägen von Fahrzeiten und Präferenzen für bestimmte Lieferorte das wirtschaftliche Potential der Standortflexibilität. Unser Branch-Price-und-Cut Framework löst realistische Krankenhausinstanzen mit bis zu 120 Behandlungen optimal und zeigt, dass das Branchen auf Zeitfenstern besser funktioniert, als das Hinzufügen von kombinatorischen Schnittebenen.

Acknowledgements

This work would not have been possible without the continuous support of my supervisor Prof. Dr. Rainer Kolisch. I would like to thank him for encouraging my research and for allowing me to grow as a scientist. I am very grateful for his continuous and proactive support of international activities, such as PhD schools and conference participations, and his flexibility and trust in my abilities when I left university and joined industry before finishing my PhD.

I would like to thank Dr. Markus Frey who has been a fantastic mentor over the last seven years, not only enhancing my academic endeavor by being co-author to both of my papers, but also by helping me to see many things in life from a different angle. I also want to express my sincere gratitude to Prof. Guy Desaulniers for warmly welcoming me to Montréal twice, and for his support during the second part of this dissertation. The structure that Guy's expertise provided helped achieving so much in relatively little time. Further, I would like to thank my committee members Prof. Dr. Maximilian Schiffer, my second reviewer, and Prof. Dr. Martin Grunow, the chair of the examination board, for serving as my committee.

I want to express my gratitude to Dr. Ferdinand Kiermaier who showed me the impact that operations research can have in industry. I want to thank Dr. Christian Ruf who shared office with me for almost five years. Christian was always open to discussing ideas, and he taught me so much about advanced topics of programming in general and Java in particular. My thanks also go to Dr. Stephen Starck for spending huge amounts of time revising my papers in a very interactive, collaborative and pleasant way.

Furthermore, I would like to thank Prof. Dr. Jens Brunner and Prof. Dr. Raik Stolletz who influenced my operations research carrier early on. Over ten years ago, I got a first glimpse of how much fun operations research could be when participating in one of Jens' projects. By being part of one of Raik's projects at the Technical University of Denmark, I got a much broader understanding of mathematical modeling, which paved the way for my theses culminating in this very document.

To my former and current colleagues at the Department of Operation and Supply Chain Management at the Technical University of Munich (in alphabetic order): Pia Ammann, Christopher Bersch, Dr. Claus Brech, Giacomo Dall'Olio, Dr. Martin Fink, Dr. Thomas Flieder, Prof. Dr. Pirmin Fontaine, Prof. Dr. Andreas Fügener, Dr. Daniel Gartner, Gregor Godbersen, Thomas Hagspihl, Christian Jost, Prof. Dr. John J. Kanet, Gina Lambert, Tobias Lierberum, Dr. Anulark Naber, Dr. Maximilian Pohl, Dr. Sebastian Schiffels, Christine Steinberger, Dr. Martin Tritschler, and Hendrik Weber - thank you for your support and friendship over the last few years.

Additionally, I want to thank Dr. Florence Gauzy Krieger and the Scientific Coordination Office of the Bavarian Research Alliance for the financial support of both of my research stays in Montréal. Mrs. Gauzy was always very helpful and goal-oriented and without her and the financial support, my second stay in Montréal would not have been possible.

Finally, I want to express my utmost gratitude to my wife, Verena Jungwirth, and my parents, Heike and Axel Döge, for their motivation and support during the last years. I know what I put Verena through the last couple of months, and words cannot express how grateful I am for having her.

Munich, May 2020

Table of Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

ALNS	Adaptive large neighborhood search
BPC	Branch-price-and-cut
CG	Column generation
CS	Charging station
E-VRP	Electric vehicle routing problem
ESPPRC	Elementary shortest path problem with resource-constraints
G-VRP	Green vehicle routing problem
GLS	Guided local seach
GVRP	Generalized vehicle routing problem
GVRPTW	Generalized vehicle routing problem with time windows
LAP	Location allocation problem
LCC	Location-capacity cuts
LB	Lower bound
lm-SRC	Limited-memory subset-row cut
LRP	Location routing problem
MDVRP	Multi-depot vehicle routing problem
MIP	Mixed-integer program
PLRP	Periodic location routing problem
PP	Pricing problem (subproblem)
RC	Reduced cost
RCPSP	Resource-constrained project scheduling program
RMP	Restricted master problem
SAH	Sequential allocation heuristic
SRC	Subset-row cut
ThSRP	Therapist scheduling and routing problem
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
UB	Upper bound

1 Introduction

1.1. Motivation

The shortage of qualified personnel is a crucial challenge for health care systems worldwide (WHO, 2016). In aging societies, like most western and some Asian ones, demand for health services increases faster than its supply. Employing available health workers as efficient as possible is the only short-term solution to mitigate the negative effects of understaffed health care systems.

In this context, we study the hospital-wide therapist scheduling and routing problem (ThSRP), which we model as a new variant of the well-known vehicle routing problem (VRP). The ThSRP is a daily scheduling problem arising at almost every hospital (Gartner et al., 2018), which can be classified as offline operational resource capacity planning according to the health care planing matrix by Hans et al. (2012).

On a daily basis, a hospital planner assigns therapists to treatments, treatments to rooms, and start times to treatments. The therapists have different shift patterns (morning, evening, or regular shift) and levels of qualification, which define the type of treatment they are available and qualified for. Treatments have a known duration and a start time window in which service must begin. If a patient receives multiple treatments throughout one day, precedence relations may exist between them, i.e. the preceding treatment must be finished before the succeeding treatment can start. For most patients, multiple possible treatment locations exist, i.e. patients can receive service at a central therapy center (TC) or at the ward in which they are staying. As only a limited number of treatments can be performed in parallel at the TC, the hospital planner must ensure location availability at all times.

Current hospital planning is a manual and time consuming task leading to unsatisfactory results. Every morning before the treatments start, a hospital planner is spending roughly an hour generating the schedule for the ongoing day. For the hospital planner, it is already difficult to generate feasible solutions and frequently the resulting schedules leave patients unserved.

To facilitate planning, we present two solution approaches: (a) a hybrid adaptive large neighborhood search (ALNS), and (b) an exact branch-price-and-cut (BPC) algorithm. While in traditional scheduling approaches the routing of the therapists would be an indirect result, we are addressing the routing decisions explicitly, which has practical as well as theoretical advantages. In large hospitals, therapists are spending a considerable part of their working times traveling between treatment locations. Minimizing traveling gives more time to treat patients, which could be used on a tactical level to increase the number of appointments scheduled per day.

Thus, routing is central for the practical application, however it complicates solving the problem compared to traditional scheduling approaches not involving routing decisions. To the best of our knowledge, only Gartner et al. (2018) have studied the ThSRP prior to this work. The authors model the ThSRP as a multi-mode resource-constrained project scheduling problem (MMRCPSP) in which the therapist routing constraints pose substantial challenges. Gartner et al. (2018) solve the problem by employing a cutting plane algorithm relaxing the therapist routing constraints and adding cuts once a violation of the latter constraints has been identified. The approach worked well for their instances since 30 minute planning intervals are used and only a limited number of cutting planes had to be generated.

In our case however, we will plan on a more granular level of 5 minute intervals and we will account for additional hospital types for which the routing decisions become more relevant. Therefore, we model the ThSRP as a VRP which we call the vehicle routing problem with time windows and flexible delivery locations (VRPTW-FL). The VRPTW-FL is a variant of the vehicle routing problem with time windows (VRPTW) with heterogeneous fleet, operations and resource synchronization and flexible service locations. Precedence relations between treatments are considered by operations synchronization, i.e. the service of one treatment must be finished before another treatment can start. The resource synchronization is needed to model the time-dependent location capacity that acts as a renewable resource.

The central novelty of our work is to address flexible service locations and capacities for the location simultaneously in one routing problem. Individually, both aspects have been addressed in the literature recently. Location capacities that influence the routing decisions have, to the best of our knowledge, only been studied in the contexts of electric and alternative fuel VRPs (cf. Bruglieri et al., 2019; Froger et al., 2019). Charging and fuel stations are capacitated and allow only a limited number of vehicles to recharge at a time. However, service sites of customers are not capacitated, neither do multiple service locations for customers exist.

Multiple service locations without capacities are studied in the contexts of parcel and last-mile deliveries in which multiple delivery points per customer could exist. These delivery points could be the customer's home, the trunk of the customer's car or a public locker. The applications are either modeled as a generalized VRP with time windows (GVRPTW) (cf. Yuan et al., 2020) or as a VRP with roaming delivery locations (cf. Reyes et al., 2017; Ozbaygin et al., 2017). The difference between the two formulations is that in the GVRPTW the time windows coincide for different service locations of the same customer while in the VRP with roaming delivery locations the time windows of the same customer are disjunct.

1.2. Dissertation overview

This thesis is divided into two main parts based on the methodology proposed to solve the VRPTW-FL. In Chapter 2, we develop the hybrid ALNS, and in Chapter 3, we develop the BPC algorithm. In the following, we provide a chapter-wise overview of this thesis and highlight its contributions.

Chapter 2: The Vehicle Routing Problem with Time Windows and Flexible Delivery Locations. In this part, we differentiate the VRPTW-FL from routing problems involving location decisions. We discuss the underlying graph structure in detail and develop a compact formulation to mathematically describe the problem. We propose a hybrid metaheuristic based on 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. In the metaheuristic phase, we extend the self-adaptiveness of the ALNS by allowing feasibility violations. These violations are penalized in the objective function, and the penalty weights are dynamically adjusted following a GLS approach. In our computational study, we assess the ALNS framework 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. The main contributions of Chapter 2 are threefold: (1) We introduce the VRPTW-FL, a new variant of the VRP, in which the time-dependent capacity of service locations influences the routing decisions of the vehicles. (2) We extend the self-adaptiveness of the ALNS by a GLS, which guides the algorithm faster to particularly good regions of the solution space. We show the benefit of allowing location flexibility by employing a variety of performance metrics.

Chapter 3: Exact Branch-Price-and-Cut for a Hospital Therapist Scheduling Problem with Flexible Service Locations and Time-dependent Location Capacity. Chapter 3 is stronger motivated by the underlying hospital planning problem and approaches the VRPTW-FL from the perspective of VRPs with synchronization constraints. While Chapter 2 focused on the flexible delivery locations, the crucial part of developing an exact solution approach for the VRPTW-FL is to properly address the time-dependent location capacity and the precedence relations. Recently, location capacity has been addressed by synchronization in the contexts of green-VRPs and electric-VRPs (Bruglieri et al., 2019; Froger et al., 2019), and precedence relations are naturally modeled by synchronizations. We solve the ThSRP by BPC in which we first relax the synchronization constraints and enforce these at a later stage by either branching on start time windows or by adding combinatorial Benders cuts. The main contributions of Chapter 3 are threefold: (1) We develop an exact BPC algorithm for the ThSRP to solve realistic hospital instances. The algorithm can be used to derive better schedules with less manual work for hospital planners. (2) We branch on the time windows of the treatments to alter start times such that a detected violation cannot occur in subsequent branches. (3) We demonstrate that branching can be a valid alternative to adding cutting planes when addressing synchronization constraints in a BPC framework.

2 The Vehicle Routing Problem with Time Windows and Flexible Delivery Locations

2.1. 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 §2.2 with an overview of related work focusing on VRPs incorporating location decisions. In §2.3, 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 §2.4, we present the hybrid meta-heuristic procedure used to solve the problem. We provide evidence of our algorithm's capabilities in §2.5 and conclude in §2.6.

2.2. Related work

The VRP and its extensions have been studied extensively in the literature. Textbooks include Toth and Vigo (2002a, 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. (2020).

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 2.2.1 is devoted to LRPs and their extensions, and Subsection 2.2.2 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.

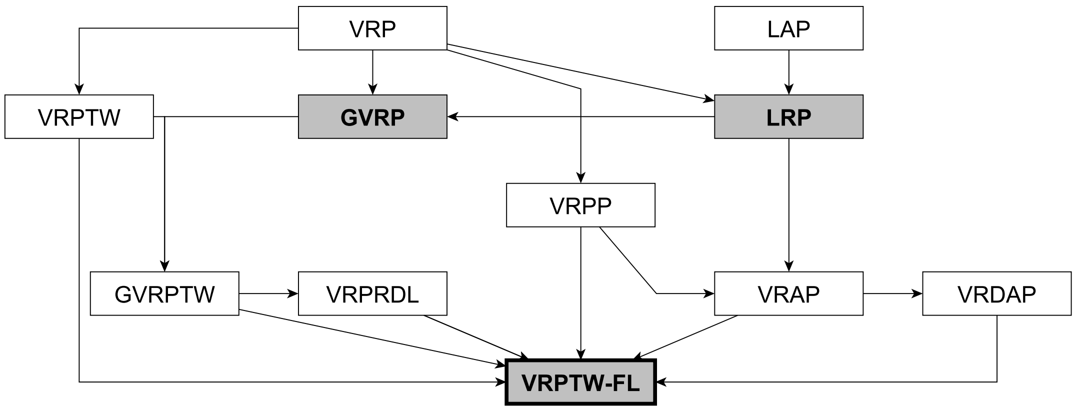


Figure 1 Connections between location and/or routing problems. The VRPTW-FL generalizes all of them. The most relevant problems are extensions of the LRP and the GVRP (gray boxes). However, the VRPTW-FL can also be seen as an extension of the VRPTW.

2.2.1. 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 2.3.4.

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 2.3.4).

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.

2.2.2. 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 2.3.4).

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. In the context of last-mile deliveries, Yuan et al. (2020) studied the single vehicle case of the GVRPTW and proposed a branch-and-cut algorithm to solve the problem.

2.3. Model development

In this section, we develop a mathematical model and discuss the underlying graph structure. Section 2.3.1 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 2.3.2, we detail the graph structure of the VRPTW-FL and demonstrate its differences from the graph of the classic VRPTW. Section 2.3.3 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 2.3.4.

2.3.1. 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 $c_{i,l}^{\text{location}} \geq 0$ 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 $c_{l,r}^{\text{travel}}$ 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 $\mathcal{I}_k \subseteq \mathcal{I}$.

2.3.2. 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.

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,

[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.

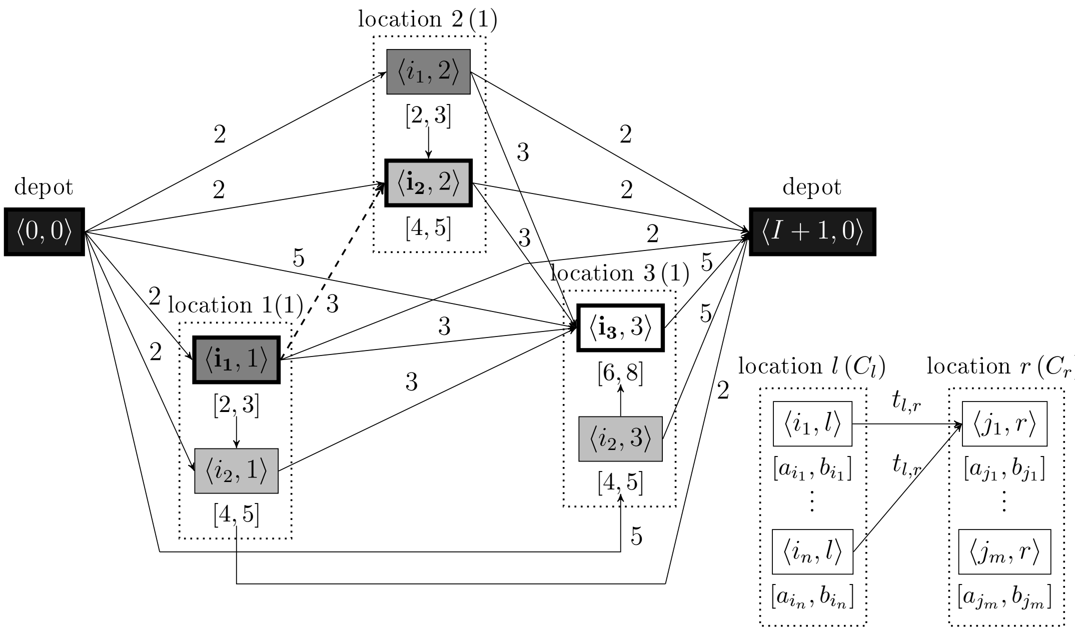


Figure 2 Routing network example: VRPTW vs. VRPTW-FL. Dotted boxes denote locations and include all customers that can be served at this location. Nodes belonging to the same customer are printed in the same color. Bold boxes denote that this location is the customer's preferred location. The dashed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

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 $c^{\text{location}}_{i,l}$. Then, a VRPTW-FL instance is uniquely given by tuple $\langle \mathcal{K}, \mathcal{I}, \mathcal{V}, \eta\rangle$ and Theorem 2.1 holds.

Theorem 2.1 *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 $c^{\text{location}}_{i,l}$ 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.

2.3.3. 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 (2.1)-(2.11):

$$\min \sum_{k\in\mathcal{K}} \sum_{\langle i,l\rangle\in\mathcal{V}_k\setminus\langle I+1,0\rangle} \sum_{\langle j,r\rangle\in\delta_k^+(i,l)} \left(c_{l,r}^{\text{travel}} + c_{j,r}^{\text{location}}\right)\cdot x_{i,l,j,r,k} \tag{2.1}$$

subject to

$$\sum_{k\in\mathcal{K}_i} \sum_{\langle i,l\rangle\in\mathcal{V}_k} \sum_{\langle j,r\rangle\in\delta_k^+(i,l)} x_{i,l,j,r,k} = 1 \qquad \forall\, i\in\mathcal{I} \tag{2.2}$$

$$\sum_{\langle j,r\rangle\in\delta_k^+(0,0)} x_{0,0,j,r,k} = 1 \qquad \forall\, k\in\mathcal{K} \tag{2.3}$$

$$\sum_{\langle i,l\rangle\in\delta_k^-(j,r)} x_{i,l,j,r,k} - \sum_{\langle i,l\rangle\in\delta_k^+(j,r)} x_{j,r,i,l,k} = 0 \qquad \forall\, k\in\mathcal{K}, \langle j,r\rangle\in\mathcal{V}_k \tag{2.4}$$

$$\sum_{\langle i,l\rangle\in\delta_k^-(I+1,0)} x_{i,l,I+1,0,k} = 1 \qquad \forall\, k\in\mathcal{K} \tag{2.5}$$

$$a_i \le T_{i,l,k} \le b_i \qquad \forall\, k\in\mathcal{K}, \langle i,l\rangle\in\mathcal{V}_k \tag{2.6}$$

$$x_{i,l,j,r,k}\cdot\left(T_{i,l,k}+s_i+t_{r,l}^{\text{travel}}-T_{j,r,k}\right)\le 0 \qquad \forall\, k\in\mathcal{K}, \langle i,l\rangle\in\mathcal{V}_k\setminus\langle I+1,0\rangle, \langle j,r\rangle\in\delta_k^+(i,l) \tag{2.7}$$

$$T_{i,l,k_1} + s_i + t_{i,j}^{\min} \leq T_{j,r,k_2} \qquad \forall \langle i,j \rangle \in \mathcal{P}, l \in \mathcal{L}_i, r \in \mathcal{L}_j, k_1 \in \mathcal{K}_i, k_2 \in \mathcal{K}_j \tag{2.8}$$

$$\sum_{i \in \mathcal{I}_l} \sum_{k \in \mathcal{K}_i} I(i,l,k,\tau) \leq C_l \qquad \forall\, l \in \mathcal{L}^{\text{bounded}}, \tau \in \mathcal{T}_l^{\text{bounded}} \tag{2.9}$$

$$\sum_{(i,l) \in \mathcal{V}_k} q_i \sum_{(j,r) \in \delta_k^+(i,l)} x_{i,l,j,r,k} \leq Q_k \qquad \forall\, k \in \mathcal{K} \tag{2.10}$$

$$x_{i,l,j,r,k} \in \{0,1\} \qquad \forall\, k \in \mathcal{K}, \langle i,l \rangle \in \mathcal{V}_k, \langle j,r \rangle \in \delta_k^+(i,l) \tag{2.11}$$

$$T_{i,l,k} \geq 0 \qquad \forall\, k \in \mathcal{K}, \langle i,l \rangle \in \mathcal{V}_k \tag{2.12}$$

Objective function (2.1) minimizes the sum of travel and location costs. The VRPTW-FL's constraint set can be divided into three parts:

Tour constraints (2.2)-(2.5): Constraints (2.2) ensure that every customer is served exactly once. Constraints (2.3)-(2.5) define the tour of each vehicle: constraints (2.3) and (2.5) impose a tour start and end at the depot, while constraints (2.4) are the flow conservation constraints.

Scheduling constraints (2.6)-(2.8): Constraints (2.6) set the start times for serving customer i, while constraints (2.7) 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 (2.8) ensure the precedence relations.

Location and vehicle capacity constraints (2.9)-(2.10): Constraints (2.9) bound the number of customers served in time interval τ at location l. Constraints (2.10) ensure that a vehicle k cannot satisfy more customer demand than its capacity limit Q_k. The variable domains are given in constraints (2.11) and (2.12).

Model formulation (2.1)-(2.11) is nonlinear due to constraints (2.7) and (2.9). Constraints (2.7) can be linearized following the approach in Desaulniers et al. (2014). However, for the linearization of constraints (2.9), 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.

2.3.4. 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 2.3.2 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 *<u>m</u>ulti-<u>d</u>epot 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.

2.4. Solution methodology

The VRPTW-FL cannot be described with a reasonable number of linear constraints (cf. §2.3.3), 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 §2.4.1. In §2.4.2, we detail the construction heuristic, including the backtracking mechanism. In §2.4.3, 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 §2.4.4, we provide implementation details focusing on preprocessing and parameter optimization.

2.4.1. 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 (2.1), then our heuristic works on the modified objective function

$$\min f^{\text{mod}}(s) = f(s) + \lambda \cdot \sum_{i \in \mathcal{I}} \left(p_i^{\text{na}} \cdot I_i^{\text{na}}(s) + p_i^{\text{tw}} \cdot I_i^{\text{tw}}(s) + p_i^{\text{pred}} \cdot I_i^{\text{pred}}(s) \right), \quad (2.13)$$

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 §2.4.3 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

[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.

$n \in [\underline{n}_h, \bar{n}_h]$ 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 §2.4.3 and §2.4.3, 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 §2.4.3.

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 §2.4.3 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.

2.4.2. 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.

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

Algorithm 1 Hybrid Adaptive Large Neighborhood Search

```
input: initial solution $s^{\text{init}}$ with objective $f^{\text{mod}}(s^{\text{init}})$          (see §2.4.2)
 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 §2.4.3 and §2.4.3)
 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 §2.4.3)
12:     end for
13:     update objective penalty terms $p^{\text{uns}}$, $p^{\text{tw}}$ and $p^{\text{pred}}$          (see §2.4.3)
14: end while
return: $s^{\text{best}}$
```

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 $0 \le m_k \le n_k$ in route r_k we have

$$T_{m_k-1} + s_{i_{m_k-1}} + t^{\text{travel}}_{l_{m_k-1}, l_{m_k}} \le T_{m_k}. \tag{2.14}$$

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 (2.14) 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 (2.13) where the penalty values (costs) for each type of violation are fixed and equal for each customer.

$$\min f^{\text{simple}}(s) = f(s)+\lambda\cdot\sum_{i\in\mathcal{I}}\left(c^{\text{na}}\cdot I_i^{\text{na}}(s)+c^{\text{tw}}\cdot I_i^{\text{tw}}(s)+c^{\text{pred}}\cdot I_i^{\text{pred}}(s)\right). \tag{2.15}$$

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:

$$\Delta g_\kappa(i,\mathcal{R}) = g_\kappa(i,\mathcal{R}) - g_1(i,\mathcal{R}). \tag{2.16}$$

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 (2.14) holds or because no location is available for customer i. To provide a repair mechanism, we implement a backtracking-branching procedure.

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 $\mathcal{I}^{\mathrm{na}}$, 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 $i \in \mathcal{I}^{\mathrm{na}}$ 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 $\mathcal{I}^{\mathrm{na}}$ 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 $\mathrm{pred}(s_h)$, the predecessor of s_h. To proceed to another solution from $\mathrm{pred}(s_h)$, 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 $\mathcal{F}(\mathrm{pred}(s_h))$ of the partial solution $\mathrm{pred}(s_h)$ (lines 17-20). The next insertion will then be the best customer-route combination with respect to regret measure (2.16) such that the corresponding customer-location start time route tuple is not contained on forbidden list $\mathcal{F}(\mathrm{pred}(s_h))$, i.e. $(i^*, r_k^*) = \arg\max_{i \in \mathcal{I}^{\mathrm{na}}} \big\{\Delta g_\kappa(i, \mathcal{R}) \mid (i, l(g_\kappa(i, \mathcal{R})), T(g_\kappa(i, \mathcal{R})), \mathcal{R}) \notin \mathcal{F}(\mathrm{pred}(s_h))\big\}$.

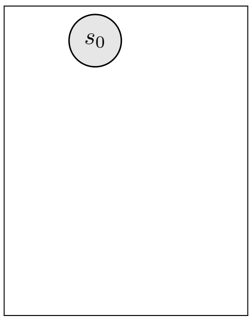
If again no feasible successor exists, i.e. the regret measure is 0 for at least one customer, we return to $\mathrm{pred}(s_h)$'s predecessor, for which we forbid the corresponding customer-location start time route tuple which would lead to $\mathrm{pred}(s_h)$ again. The procedure generates a search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

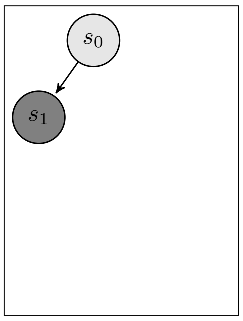
Algorithm 2 CONSTRUCTION PHASE

```
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 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: $s^{\text{best}}$
```

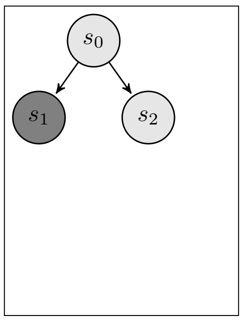
During our first computational tests of the construction heuristic, we made the following observation: Returning to the direct predecessor of an infeasible partial solution does generally not correct the solution as desired, especially if only a few customers are left to insert. Many iterations of backtracking 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



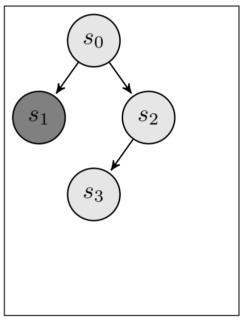
(a) Partial solution s_0



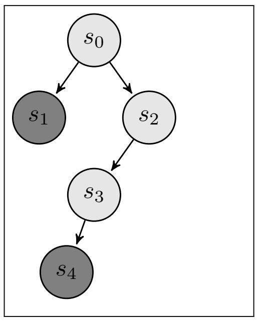
(b) Proceed to partial solution s_1, in which added tuple (i_1, l_1, T_1, r_{k_1}) leads to infeasibility



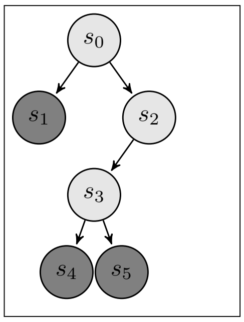
(c) Backtrack to node s_0 and set $\mathcal{F}(s_0) = \{(i_1, l_1, T_1, r_{k_1})\}$; proceed to partial solution s_2 with added tuple (i_2, l_2, T_2, r_{k_2})



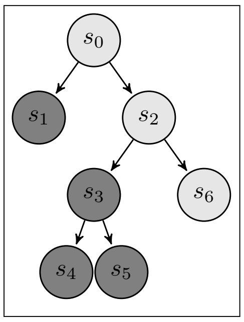
(d) Proceed to partial solution s_3 with added tuple (i_3, l_3, T_3, r_{k_3})



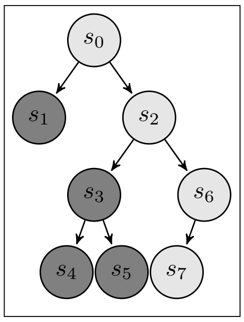
(e) Proceed to partial solution s_4, in which added tuple (i_4, l_4, T_4, r_{k_4}) leads to infeasibility



(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

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.

2.4.3. 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.

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:

$$\rho_j = \frac{w_j}{\sum_{i=1}^{|\Omega|} w_i} \tag{2.17}$$

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 (2.18)). The counter θ_j measures how often the operator has been applied.

$$w_j^{\text{updated}} = w_j \cdot (1 - r) + r \cdot \frac{\pi_j}{\theta_j} \tag{2.18}$$

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

Update objective penalty terms

In the augmented cost function (2.13), 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. $p_i^{\text{na}} = p_i^{\text{tw}} = p_i^{\text{pred}} = 0 \quad \forall i \in \mathcal{I}$. 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 (2.19), where s_h is the current solution at iteration h.

$$u(s_h, f_j) = I_j(s_h) \cdot \frac{c_j}{1 + p_j} \tag{2.19}$$

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. §2.4.3).

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 §2.3.2), all operators are also applicable for the VRPTW. The effectiveness of the procedures will be shown in the computational study in §2.5.3. 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:

$$D^{\text{time}}(i,j) = \frac{T}{\left(\alpha_1 \cdot \bar{T}_{i,j} + \alpha_2 \cdot |T_i - T_j|\right)}, \tag{2.20}$$

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

$$\bar{T}_{i,j} = \left| \frac{a_i + b_i}{2} - \frac{a_j + b_j}{2} \right|, \tag{2.21}$$

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 (2.22)). 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 ($\min\{|\mathcal{L}_i|, |\mathcal{L}_j|\}$).

$$D^{\text{loc}}(i,j) = \frac{|\mathcal{L}_i \cap \mathcal{L}_j|}{\min\{|\mathcal{L}_i|, |\mathcal{L}_j|\}} \tag{2.22}$$

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

$$D^{\text{loc,time}}(i,j) = \beta_1 \cdot D^{\text{loc}}(i,j) + \beta_2 \cdot D^{\text{time}}(i,j). \tag{2.23}$$

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:

$$\min \sum_{i=1}^{k} \sum_{j \in \mathcal{S}_i} (x_j - \mu_i)^2 . \tag{2.24}$$

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 (2.25):

$$\min \sum_{i=1}^{k} \sum_{j \in \mathcal{S}_i} \left(t_{l_j, l_j^{\text{center}}} \right)^2 . \tag{2.25}$$

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.

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 (2.16) in §2.4.2).

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.

2.5. Computational study

In this section, we investigate the performance of our algorithm. In particular, we describe in §2.5.1 how the data used in our computational experiments has been generated, and in §2.5.2 we detail how we adjusted the parameters of our algorithm. We evaluate our newly introduced algorithm features in §2.5.3. 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 §2.5.4. 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 §2.5.5, 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.

2.5.1. 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 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.

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

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.

2.5.2. 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 §2.5.3 and §2.5.4 are stated in Appendix C.

2.5.3. 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.

Value of backtracking

Our backtracking mechanism adds a look-ahead perspective to the construction heuristic to alter unsatisfactory decisions during the insertion process (cf. §2.4.2). 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 $\kappa = \{2, 3, \ldots, 6\}$. 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 2. 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%, 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.

Table 2 Comparison of construction heuristic with and without backtracking. For each variant, the objective function value (Equation (2.15)), 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				
			$f(s^*)$	$\lvert I^{\mathrm{na}}(s^*)\rvert$	$\lvert I^{\mathrm{pred}}(s^*)\rvert$	n^{feas}	$f(s^*)$	$\lvert I^{\mathrm{na}}(s^*)\rvert$	$\lvert I^{\mathrm{pred}}(s^*)\rvert$	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%	

Value of ALNS+GLS

We tested a standard ALNS working on objective function (2.15) against an ALNS working on objective function (2.13), 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 solutions even in later iterations.

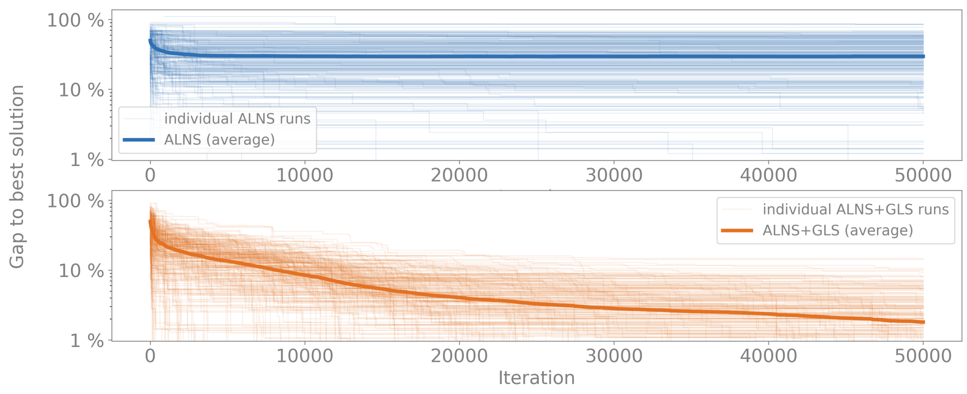


Figure 4 Comparison of ALNS (above, blue lines) and ALNS+GLS (below, orange lines). Thin lines represent the percentage gap to the best feasible solution found over all runs. The bold lines represent the average percentage gap.

Table 3 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 $n = 50,000$.

Value of new operators

To specifically tackle the underlying problem structure, we have developed several new neighborhoods (cf. §2.4.3). 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

Table 3 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

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.

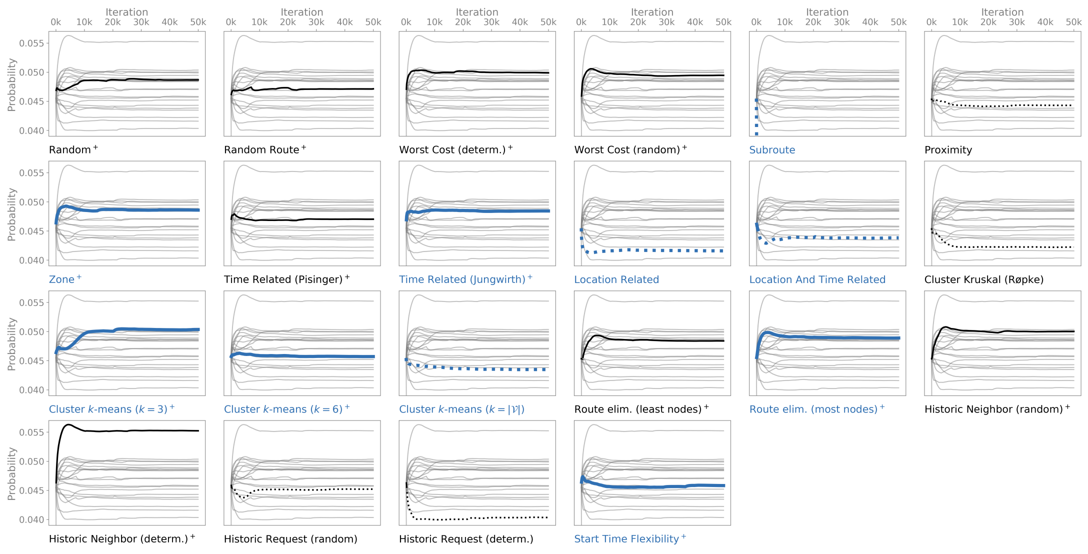


Figure 5 Probabilities of selecting the destroy operators over 50, 000 iterations. Each subplot displays the average probability of an operator over the 90 instances. Black thin lines represent operators from the literature. Blue thick lines represent our operators. Light gray lines represent the operators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

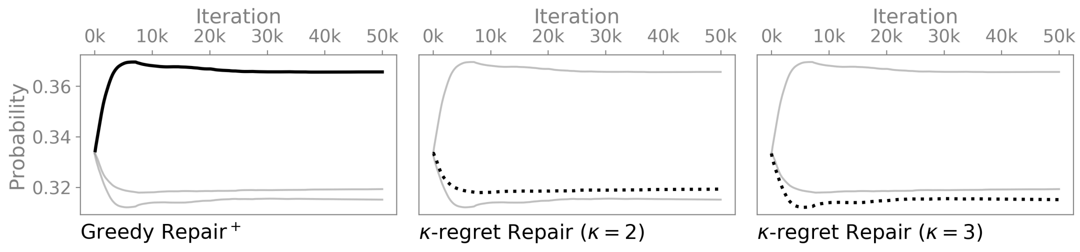


Figure 6 Probabilities of selecting the repair operators over 50, 000 iterations. Each subplot displays the average probability of an operator over the 90 instances. Black thin lines represent operators from the literature. Light gray lines represent the operators which are not the focus of the subplot.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

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 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 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% ($+2.27\%$).

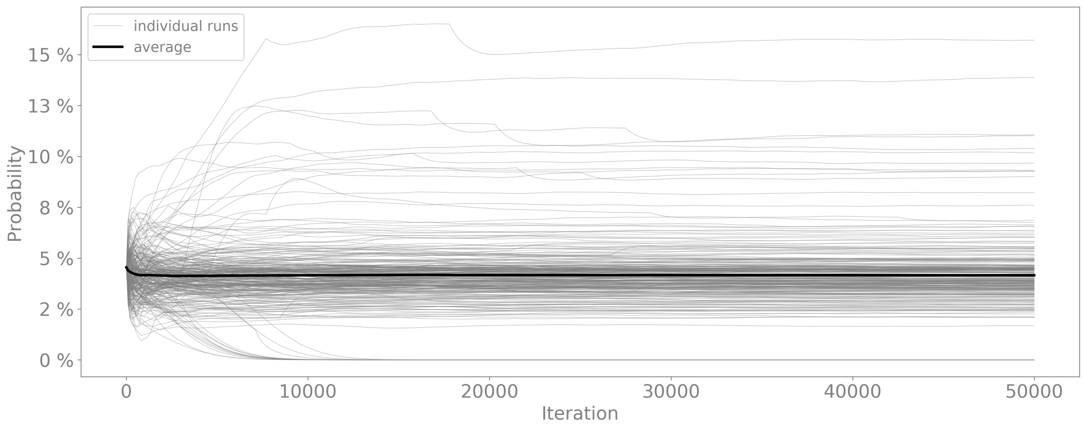


Figure 7 Distribution of the probability developments over time for the location-related destroy. Thin gray lines represent the individual runs, while the bold black line represents the average over all runs.

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 4.

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.

2.5.4. 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.

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

Table 4 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 (2.15)), 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%

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.

The results for the SAH and our ALNS are given in Table 5. 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.

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 6 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

Table 5 Comparison of hospital planning and ALNS. For each variant, the objective function value (Equation (2.15)), 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	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 6 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.

$\|\mathcal{I}\|$	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

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.

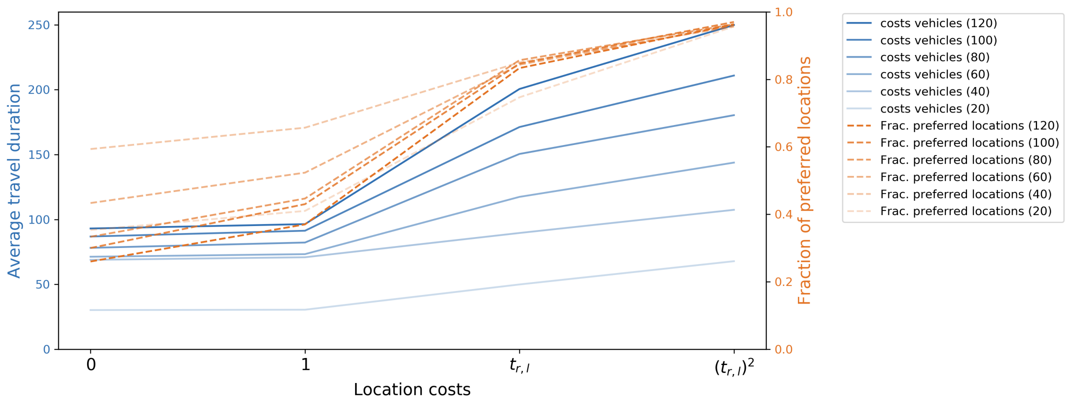


Figure 8 Travel costs of therapists (vehicles) in blue and fraction of treatments in preferred locations in orange for different location costs (0, 1, $t_{r,l}$, $(t_{r,l})^2$). Each line groups the results for different instances sizes (20, 40, . . . , 120 customers).

2.5.5. 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 §2.5.1. The results for the 29 instances of the first class with relatively narrow time windows is given in Table 7. 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 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.

2.6. 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

Table 7 Performance of our ALNS on Solomon benchmark instances. The *type*s 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

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.

3 Exact Branch-Price-and-Cut for a Hospital Therapist Scheduling Problem with Flexible Service Locations and Time-dependent Location Capacity

3.1. Introduction

The shortage of qualified personnel is a crucial challenge for health care systems worldwide (WHO, 2016). In aging societies, like most western and some Asian ones, demand for health services increases faster than its supply. Employing available health workers as efficiently as possible is the only short-term solution to mitigate the negative effects of understaffed health care systems.

In this context, we study the hospital-wide therapist scheduling and routing problem (ThSRP), a daily scheduling problem arising at almost every hospital (Gartner et al., 2018). According to the health care planning matrix by Hans et al. (2012) the ThSRP can be classified as an offline operational resource capacity planning problem. On a daily basis, a hospital planner assigns therapists to treatments, treatments to rooms, and start times to treatments. The therapists have different shift patterns (morning, evening, or regular shift) and levels of qualification. Shift and qualification define the type of treatment therapists are available and qualified for. Treatments have a known duration and a start time window, in which service must begin. If a patient receives multiple treatments in the same day, precedence relations may exist between them, i.e. the preceding treatment must be finished before the succeeding treatment can start. For most patients, multiple treatment locations exist, i.e. patients can receive service at a central therapy center (TC) or at the ward in which they are staying. There are two noteworthy exceptions: outpatients can only be treated at the TC, and bedridden patients can only be treated at the ward since they must not be moved.

TCs are capacitated, i.e. only a limited number of patients can be treated at a time. Thus, for the TCs the hospital planner must ensure location availability at all times. However, wards can be seen as uncapacitated as no patient will be scheduled to another patient's ward room for treatment.

Current hospital planning is a manual and time consuming task leading to unsatisfactory results. Every morning before the treatments start, a hospital planner is spending roughly an hour generating the schedule for the ongoing day. For the hospital planner, it is already difficult to generate feasible solutions and frequently the resulting schedules leave some patients unserved. To facilitate planning, we develop an exact algorithm to solve realistic hospital instances. While in traditional scheduling approaches, the routing of the therapists would be an indirect result, we are addressing the routing decisions explicitly, which has practical as well as theoretical advantages. In large hospitals, therapists are spending a considerable part of their working time traveling between treatment locations. Minimizing traveling gives more time to treat patients, which can be used on a tactical level to increase the number of appointments scheduled per day. However, a hospital might have preferences for treating patients in specific locations and these preferences can conflict with minimizing traveling. Therefore, we associate a certain cost with treating a patient in a specific location, and minimize these location costs together with the traveling.

We model the ThSRP as a Vehicle Routing Problem with Time Windows (VRPTW), heterogeneous fleet, operations and resource synchronization, and flexible service locations. Precedence relations between treatments are considered by operations synchronization, and resource synchronization is needed to model the time-dependent location capacity that acts as a renewable resource. As this paper focuses on routing, we will occasionally fall back on the VRP terminology, i.e. instead of treatment and therapist, we use customer and vehicle, respectively. For other applications of routing in hospitals, see e.g., Beaudry et al. (2010); Hanne et al. (2009), and Schmid and Doerner (2014).

We solve the ThSRP by branch-price-and-cut (BPC), in which we address the synchronization constraints using one of the following two alternatives: (1) We branch on the time windows of the treatments to alter the possible start times such that a violation cannot occur in subsequent branches, or (2) we add combinatorial Benders cuts to forbid the reoccurrence of the current solution by specifying which combinations of arcs must not appear together. Branching on start time windows received little attention over the past 25 years. However as we will show, it holds the potential of being a valuable building block of solving complex routing problems. To avoid unnecessary branching steps, we use additional branching strategies to break symmetry at higher levels and we discuss strategies to correct infeasible solutions.

Our contribution is threefold: (1) We model the ThSRP as a VRPTW with flexible delivery locations and synchronization constraints, which are properties relevant to other VRP variants as well; (2) we develop an exact BPC algorithm for the ThSRP to solve realistic hospital instances, which can be used to derive better schedules with less manual work for hospital planners; and (3) we show that time window branching can be a valid alternative to adding cutting planes when addressing synchronization constraints in a BPC framework.

The remainder of this work is structured as follows. In §3.2, we give an overview of the related literature focusing on VRPs. A formal definition of the problem is presented in §3.3. We describe the details of our BPC algorithm in §3.4, including specifics of the graph structure of the pricing problem, the branching strategies, and the cuts. In §3.5, we evaluate the performance of the proposed algorithm and some of its features, and show the robustness of the method against changing inputs. We conclude in §3.6.

3.2. Literature

The literature review focuses on VRPs as our main contributions are related to routing problems. First we will give a brief overview of routing related applications in health care contexts, and then we specifically focus on the three components that distinguish our problem from traditional VRP and VRPTW: (a) flexible service location, (b) time-dependent location capacity, and (c) precedence relations. For general literature on VRPs see e.g., Toth and Vigo (2002b, 2014) and Vidal et al. (2020). A very recent overview of exact BPC methods for VRPs is given in Costa et al. (2019).

Health care related routing applications are numerous and range from transporting patients to hospitals (Nemati et al., 2016; Lim et al., 2017) over routing of bloodmobiles (Gunpinar and Centeno, 2016) to distributing pharmaceutical products to health care facilities (Kramer et al., 2019). A significant share of the heath care literature related to routing is concerned with home care applications, which are reviewed in Cissé et al. (2017) and Fikar and Hirsch (2017). Only few papers study routing in hospitals. Hanne et al. (2009) and Beaudry et al. (2010) develop a heuristic to solve a routing problem with fixed locations and dynamically arriving transportation requests. Schmid and Doerner (2014) present a metaheuristic for an integrated planning problem combining scheduling and routing of patients. The ThSRP is studied in Gartner et al. (2018) who present an insertion heuristic and an exact algorithm. The exact algorithm solves a scheduling problem after relaxing the routing constraints, and if the routing constraints are violated in the optimal solution, a cutting plane is added and the problem is solved again. To the best of our knowledge, this is the only exact algorithm for the ThSRP so far.

In the remainder of this review, we will focus on routing problems showing similarities to the ThSRP in terms of modeling and methodology. In the literature, we found two types of problems involving flexible service locations: (1) Beasley and Nascimento (1996) introduced the vehicle routing-allocation problem, in which multiple possible service locations exist for customers. These locations are uncapacitated and customers can also be left unserved. Practical applications are, e.g. routing mobile clinics in rural areas or designing postal collection routes. (2) Reyes et al. (2017) introduced the VRP with roaming delivery locations in which parcels are delivered to different locations depending on the time of the day. A branch-and-price algorithm for this problem is developed in Ozbaygin et al. (2017). Locations are not capacitated and the structure of the time windows differs from our case as different locations have distinct and non-overlapping time windows. When time windows are not distinct for different locations of the same customer, the VRP with roaming delivery locations is known as the GVRPTW. Recently, Yuan et al. (2020) developed a branch-and-cut algorithm for a GVRPTW in the context of last-mile delivery.

The stream of literature sharing most similarities with our problem is the VRP with synchronization constraints. An overview of VRPs with synchronization is given in Drexl (2012), and recent applications are e.g., Fink et al. (2019) and Liu et al. (2019).

According to the taxonomy of Drexl (2012) our time-dependent location capacities belong to resource synchronizations. Generally, tours in VRPs are independent of one another, i.e. a change *within* one tour does not affect another tour. However, when resource synchronization is present, a change within one tour could potentially change all other tours.

If location capacity is considered in the VRP literature, these capacities generally do not affect the routing decisions of the vehicles, i.e. vehicles simply wait at the location until the resource becomes available again. An example is log truck scheduling studied by Hachemi et al. (2013) and Rix et al. (2015). In this forestry application, log-loaders can only load one truck at a time, and if the loader is in use, other trucks have to wait. In the VRP with location congestion (Lam and Hentenryck, 2016), vehicles are synchronized with regard to e.g., parking lots and forklifts. Dynamic scheduling of automated guided vehicles at airports is studied by Ebben et al. (2005) with scarce resources being docks for (un)loading and parking space. If the resources are unavailable, vehicles wait at the destination, or begin their tours later.

Recently, limited availability of recharging and refueling has been studied in electric-VRPs (E-VRPs) and green VRPs (G-VRPs), cf. Keskin et al. (2019) and Froger et al. (2019) for E-VRPs, and Bruglieri et al. (2019) for G-VRPs. In Keskin et al. (2019), vehicles wait until charging becomes available. In contrast, in Bruglieri et al. (2019) and Froger et al. (2019) the time-dependent location capacity is a central part of the routing decision. The problems the latter authors study are, from the routing literature, those which are closest to the ThSRP. As in our case, it is not known in advance which tasks will be needed to be synchronized and the synchronization affects the routing decisions.

In E-VRPs and G-VRPs, refueling between two customer visits is possible. In Froger et al. (2019) (E-VRP), refueling happens at a charging station (CS) equipped with a number of charging points, and in Bruglieri et al. (2019) (G-VRP), alternative fuel stations with multiple fuel pumps can be visited. Examples of alternative fuels are e.g., methane and electricity. Froger et al. (2019) include more specifics of recharging e.g., non-linear charging functions, and vehicles do not need to recharge to full capacity when visiting as a CS.

Both works consider capacity at the level of the charging points. Each charger has a capacity of 1, i.e. exactly one vehicle can recharge at a time. However, there can be multiple charging points at a CS. In each work, two formulations are presented: one based on virtually cloning the CSs, and one based on aggregating subsets of arcs to paths. The main difference between these works and ours is that in their case, charging is capacitated and charging enables visiting additional customers. However, the service locations of the customers itself are not capacitated, neither do flexible service locations exist. In ThSRP, enforcing capacity at the service location has influence on the decision where to serve a customer. Another central difference is the importance of time windows in our case. Time windows are not considered in Froger et al. (2019), and Bruglieri et al. (2019) only discuss adding time slots at CSs, which could be reserved, however these are not time windows at service sites.

The third distinctive component of the ThSRP are the precedence constraints, which belong to operations synchronization according to the taxonomy of Drexl (2012). This type of synchronization is more common in the VRP literature than resource synchronization. It occurs e.g., in dial-a-ride problems, and in pickup-and-delivery problems (see e.g., Mitrović-Minić and Laporte, 2006; Groër et al., 2009; Hachemi et al., 2013).

Table 8 summarizes the properties of the most related routing problems. Only few papers consider flexible service locations or time-dependent location capacities. Precedence relations and time windows were either not or not much studied in these works. To the best of our knowledge, we are the first to address all four aspects combined in one problem.

Table 8 Overview of VPRs related to the ThSRP.

	Br	Fr	Be	Re/Oz	Our work
Flexible service locations			✓	✓	✓
Time-dependent location capacity	✓	✓			✓
Time windows	(✓)			✓	✓
Precedence relations					✓

(Br) Bruglieri et al. (2019); (Fr) Froger et al. (2019); (Be) Beasley and Nascimento (1996); (Re) Reyes et al. (2017); (Oz) Ozbaygin et al. (2017).

Outside of the VRP domain, operations and resource synchronization naturally occur in resource-constrained project scheduling problems (RCPSP). See Brucker et al. (1999) for an overview of RCPSPs and De Reyck et al. (1999) particularly for precedence relations. In fact, Gartner et al. (2018) model a similar ThSRP on a less granular level as a multi-mode RCPSP.

3.3. Set-covering formulation

In this part, we reformulate the compact model (2.1)-(2.11) as a set-covering model using Dantzig-Wolfe decomposition (see e.g., Lübbecke and Desrosiers, 2005). The ThSRP can formally be described as follows. A set of treatments (customers) $\mathcal{I}$ must be covered by a set of tours Ω. A binary parameter $\alpha_{i,l,r}$ indicates if treatment i in location l is part of tour r. The set of tours Ω can be broken down to $\Omega_{s,q} \subseteq \Omega$ containing only tours, which are available to therapists (vehicles) with shift type $s \in \mathcal{S}$ and at least qualification $q \in \mathcal{Q}$. Parameter $|\mathcal{K}_{s,q}|$ states the number of available therapists of shift type s and qualification q, and decision variable $\gamma_{s,q}$ holds the number of active therapists. A detailed description of how the tours are generated is given in §3.4.1.

A treatment i requires a service time s_i, and can begin in starting time window $[a_i, b_i]$ with $a_i \leq b_i$. We define i_0 and i_{n+1} to be dummy treatments for leaving and entering the depot. In the ThSRP, the depot corresponds to the break rooms, in which therapists stay when they are not on duty. Between two treatments i and j a precedence relation $(i, j) \in \mathcal{P}$ can exist, i.e. service of i must be finished before service of j can be started. Let $\mathcal{L}$ be the set of locations including the depot. Treatment i can be performed at locations $l \in \mathcal{L}_i \subseteq \mathcal{L}$. Locations $l \in \mathcal{L}^{\text{bounded}} \subseteq \mathcal{L}$ have a capacity limit of Q_l. The travel time between locations l_1 and l_2 is denoted by t_{l_1,l_2}. Binary parameter $\beta_{l,t,r}$ controls if location l is occupied at time t by tour r, and by parameter $T_{i,l,r}$ we denote the start time of treatment i in location l in tour r.

The objective is to minimize costs c_r over all tours r, which includes costs $c^{\text{travel}}_{l_1,l_2}$ for traveling between locations $l_1, l_2 \in \mathcal{L}$ and costs $c^{\text{location}}_{i,l}$ for treating i at location l. The latter is relevant to model that hospitals might have preferences for one location over the other. Decision variable λ_r is equal to 1 if tour $r \in \Omega$ is selected in the solution, and 0 otherwise. With the definitions above, the set-covering formulation of the ThSRP can be stated as:

$$\min \sum_{(s,q)\in\mathcal{S}\times\mathcal{Q}} \sum_{r\in\Omega_{s,q}} c_r \cdot \lambda_r \tag{3.1}$$

subject to

$$\sum_{(s,q)\in\mathcal{S}\times\mathcal{Q}:i\in\mathcal{I}_{s,q}} \sum_{r\in\Omega_{s,q}} \sum_{l\in\mathcal{L}_i} \alpha_{i,l,r} \cdot \lambda_r \geq 1 \quad \forall\, i \in \mathcal{I} \tag{3.2}$$

$$\sum_{r\in\Omega} \beta_{l,t,r} \cdot \lambda_r \leq Q_l \quad \forall\, l \in \mathcal{L}^{\text{bounded}}, t \in \mathcal{T}^{\text{bounded}}_l \tag{3.3}$$

$$\left(T_{i,l_i,r} + s_i + t_{l_i,l_j} - T_{j,l_j,r'}\right) \cdot \alpha_{i,l_i,r} \cdot \alpha_{j,l_j,r'} \cdot \lambda_r \cdot \lambda_{r'} \leq 0 \quad \forall\, r, r' \in \Omega, (i,j) \in \mathcal{P} \tag{3.4}$$

$$\sum_{r \in \Omega_{s,q}} \lambda_r = \gamma_{s,q} \quad \forall\, (s,q) \in \mathcal{S} \times \mathcal{Q} \tag{3.5}$$

$$\lambda_r \in \{0,1\} \quad \forall\, r \in \Omega \tag{3.6}$$

$$0 \leq \gamma_{s,q} \leq |\mathcal{K}_{s,q}| \quad \forall\, (s,q) \in \mathcal{S} \times \mathcal{Q} \tag{3.7}$$

Objective (3.1) minimizes the cost over all selected tours. Assignment constraints (3.2) ensures that each treatment i is done at least once. Location capacity constraints (3.3) guarantee for every capacitated location l and for every time point t that the capacity limit Q_l is not exceeded. The set of critical time points $t \in \mathcal{T}_l^{\text{bounded}}$ is determined in a preprocessing step. Precedence constraints (3.4) establish that preceding treatments are finished before the succeeding treatments start. Constraints (3.5) compute the number of active therapists. Lastly, the variable domains are defined in (3.6) and (3.7).

3.4. Branch-price-and-cut algorithm

To solve the set covering formulation in §3.3, we would need to generate all possible tours Ω. As generating all tours is practically impossible, column generation (CG) is used and embedded in a branch-and-cut framework to yield an exact BPC algorithm (cf. Vanderbeck and Wolsey, 1996; Barnhart et al., 1998; Desaulniers et al., 2005). The idea behind CG is to start with a reasonably small subset of tours $\Omega' \subseteq \Omega$ in a so-called restricted master problem (RMP), restricted because it contains only a subset of tours. Then, the RMP is solved and its dual solution is used to generate new promising tours in a pricing problem (PP). Since a column represents a tour, we use these terms interchangeably. The PP aims to find new tours with negative reduced cost (RC). If new tours are found (RC<0), they are added as columns to the RMP, and if no new columns are found (RC≥ 0), CG terminates and a valid lower bound for the minimization problem was found. Details on the PP are given in §3.4.1.

If CG terminates and if $\lambda_r \in \mathbb{N}_0$ for all $r \in \Omega'$, an optimal solution for the original problem has been found. However, if at least one non-integer variable $\lambda_r \notin \mathbb{N}_0$ exists, branching is needed. Traditionally, branching on the number of vehicles (therapists) and branching on arcs connecting customers (treatments) is used for VRPs (cf. Feillet, 2010). We use both strategies and introduce three additional branching strategies before branching on arcs: (a) Branching on the type of therapist performing a treatment, (b) branching on the service location of a treatment, and (c) branching on aggregated therapy arcs. The branching strategies are described in detail in §3.4.2. To tighten the dual bound, we add limited-memory subset-row cuts (lm-SRCs) introduced by Pecin et al. (2017b), and we propose location-capacity cuts (LCC) to exclude combinations of treatments that will result in a capacity violation of a specific service location. Details of the cuts are given in §3.4.4.

To solve the problem, we first relax the synchronization constraints for the location capacity (3.3) and the precedence relations (3.4) in the original formulation (3.1)-(3.7). The resulting RMP (3.8)-(3.12) is equivalent to the set covering formulation of the heterogeneous VRPTW. The difference, however, is that if we have found an integer feasible solution, we still need to check if the solution is also feasible for the synchronization constraints (3.3) and (3.4). If yes, the original problem was solved successfully, and if not, we present two approaches to eliminate infeasibilities: (a) by branching on start time windows (see §3.4.2), and (b) by adding combinatorial Benders cuts (see §3.4.4). A schematic overview of our BPC framework is displayed in Figure 9.

$$\min \sum_{(s,q)\in\mathcal{S}\times\mathcal{Q}} \sum_{r\in\Omega'_{s,q}} c_r \cdot \lambda_r \tag{3.8}$$

subject to

$$\sum_{(s,q)\in\mathcal{S}\times\mathcal{Q}:i\in\mathcal{I}_{s,q}} \sum_{r\in\Omega'_{s,q}} \sum_{l\in\mathcal{L}_i} \alpha_{i,l,r}\cdot\lambda_r \geq 1 \quad \forall\, i\in\mathcal{I} \tag{3.9}$$

$$\left(\sum_{r\in\Omega'_{s,q}} \lambda_r\right) - \gamma_{s,q} = 0 \quad \forall\,(s,q)\in\mathcal{S}\times\mathcal{Q} \tag{3.10}$$

$$\lambda_r \geq 0 \quad \forall\, r\in\Omega' \tag{3.11}$$

$$0 \leq \gamma_{s,q} \leq |\mathcal{K}_{s,q}| \quad \forall\,(s,q)\in\mathcal{S}\times\mathcal{Q} \tag{3.12}$$

Note that an alternative to relaxing the precedence constraints would be to leave the constraint in RMP, which then results in a significantly more complicated PP with linear node costs. This approach is pursued in He et al. (2019) who build on the work of Ioachim et al. (1998).

3.4.1. Pricing problem

To generate new columns for the RMP, we solve $|\mathcal{S}\times\mathcal{Q}|$ PPs, each corresponding to a specific therapist type. The PPs, also called subproblems, correspond to elementary shortest path problems with resource constraints (ESPPRC) and aim

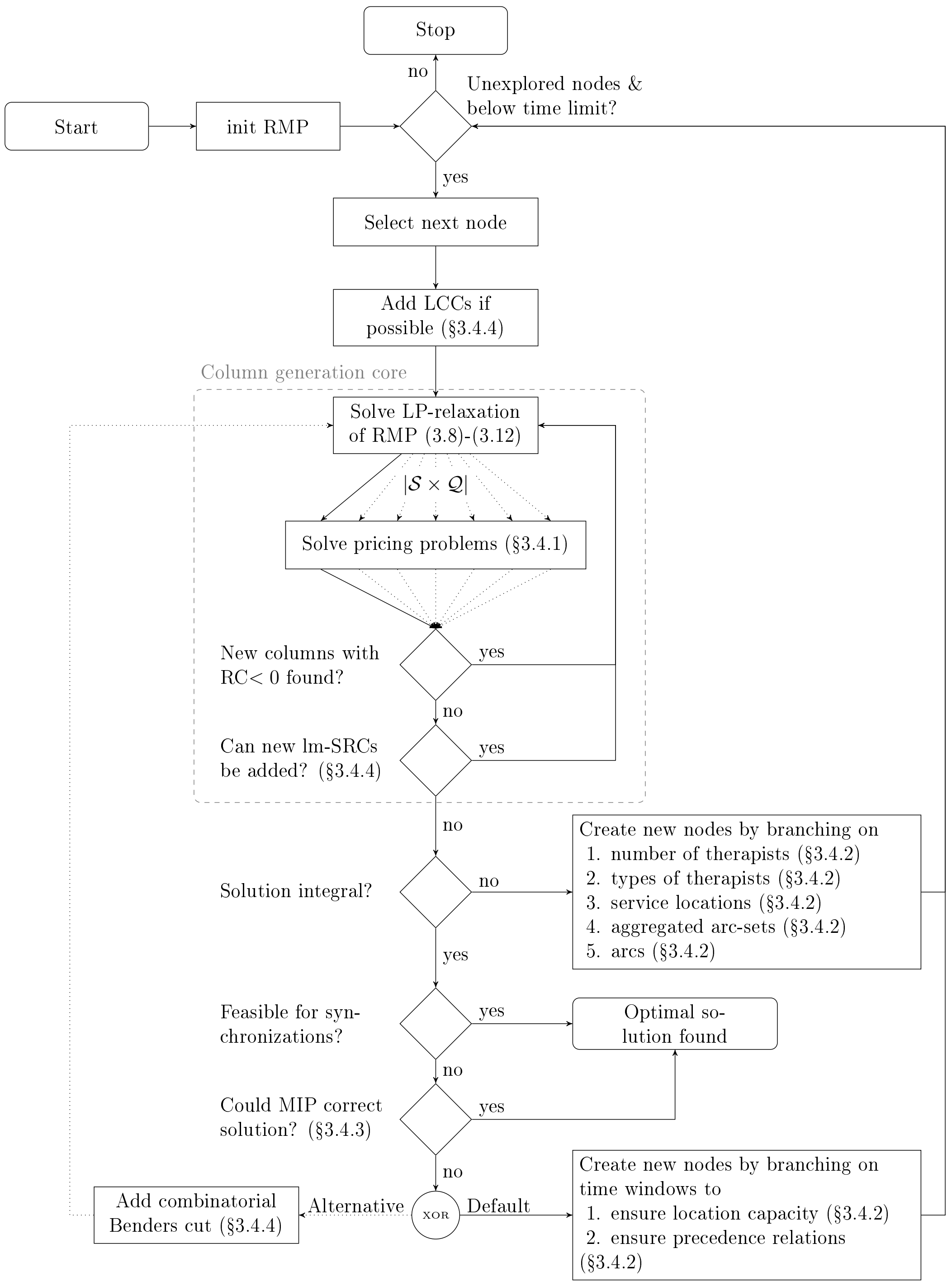

Stop
no
Unexplored nodes & below time limit?
Start
init RMP
yes
Select next node
Add LCCs if possible (§3.4.4)
Column generation core
Solve LP-relaxation of RMP (3.8)-(3.12)
$|\mathcal{S} \times \mathcal{Q}|$
Solve pricing problems (§3.4.1)
New columns with RC< 0 found?
yes
no
Can new lm-SRCs be added? (§3.4.4)
yes
no
Solution integral?
no
Create new nodes by branching on
1. number of therapists (§3.4.2)
2. types of therapists (§3.4.2)
3. service locations (§3.4.2)
4. aggregated arc-sets (§3.4.2)
5. arcs (§3.4.2)
yes
Feasible for synchronizations?
yes
Optimal solution found
no
Could MIP correct solution? (§3.4.3)
yes
no
Add combinatorial Benders cut (§3.4.4)
Alternative
XOR
Default
Create new nodes by branching on time windows to
1. ensure location capacity (§3.4.2)
2. ensure precedence relations (§3.4.2)

Figure 9 Schematic overview of our BPC framework.

at finding new feasible tours $r \in \Omega$ with negative RC. Let $\pi_i^{(3.9)}$ and $\pi_{s,q}^{(3.10)}$ be the duals associated with constraints (3.9) and (3.10) of the RMP, respectively. Then solving the ESPPRC is equivalent to the following minimization problem for a given shift type $s \in \mathcal{S}$ and qualification $q \in \mathcal{Q}$:

$$\min_{r \in \Omega_{s,q}} \bar{c}_r = c_r - \sum_{i \in \mathcal{I}_{s,q}} \sum_{l \in \mathcal{L}_i} \alpha_{i,l,r} \cdot \pi_i^{(3.9)} - \pi_{s,q}^{(3.10)} \tag{3.13}$$

The ESPPRC is modeled on graph a $G = (\mathcal{V}, \mathcal{A})$ with vertex set $\mathcal{V}$ and arc set $\mathcal{A}$. The definition of the node set $\mathcal{V}$ differs from classical VRPs (cf. Feillet et al., 2004; Irnich and Desaulniers, 2005). Instead of having only one node v_i per treatment for a customer,, we have $|\mathcal{L}_i|$ nodes $v_{i,l}$, one for each possible service location $l \in \mathcal{L}_i$ for a treatment of a customer. We define node v_0 to represent the depot and the subset of nodes without the depot is denoted by $\mathcal{V}' = \mathcal{V} \setminus \{v_0\}$. Figure 10 displays an example of how the complexity of graph G increases in our problem compared to the VRP.

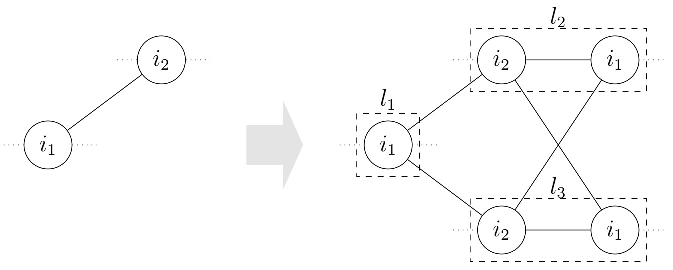


Figure 10 Arc (i_1, i_2) in the VRP (left) expands into a subgraph in the ThSRP (right).

An individual start time window is assigned to each node, i.e. even nodes representing the same treatment i might have different time windows. Since a node contains location information, time window tightening might affect nodes belonging to the same treatment i but different location l, differently, and branching on time window presented in §3.4.2 affects the time windows of the nodes rather than the ones of the treatments.

For arc set $\mathcal{A} \subseteq \mathcal{V} \times \mathcal{V}$, we have $(v_{i,l}, v_{j,r}) \in \mathcal{A}$ with $v_{i,l}, v_{j,r} \in \{\mathcal{V} : i \neq j\}$ if and only if treatment i can be serviced at location l before treatment j is served at location r. The costs $\bar{c}_{v_{i,l},v_{j,r}}$ along an arc $(v_{i,l}, v_{j,r}) \in \mathcal{A}$ are defined in (3.14). Travel cost is defined between location $l, r \in \mathcal{L}$ and costs for servicing in location $r \in \mathcal{L}$ are considered for the head node. The dual $\pi_{s,q}^{(3.10)}$ associated with the convexity constraint (3.10) is also subtracted at the depot node v_0.

$$\bar{c}_{v_{i,l},v_{j,r}} = c_{l,r}^{\text{travel}} + c_{j,r}^{\text{location}} - \pi_i^{(3.9)} \tag{3.14}$$

We solve the ESPPRC by using a dynamic programming labeling algorithm starting from the source v_0 and iteratively extends partial paths to new nodes $v_{i,l} \in \mathcal{V}$ until it reaches the sink $v_{n+1} = v_0$. Relevant information of the partial paths is stored in label L_v associated with the nodes of the graph. Specifically, L_v contains the following information: the parent label id, the cumulated RC, the earliest arrival time at the node, and a binary vectors stating which treatments can still be done. In order to not enumerating all paths, a dominance criterion is applied to discard non-useful labels. For further details on labelling algorithms, we refer the reader to Feillet et al. (2004), Irnich and Desaulniers (2005), and Irnich (2008).

Since each PP is called frequently, we apply the following acceleration techniques: time window tightening (Desrochers et al., 1992; Kontoravdis and Bard, 1995), bidirectional search (Righini and Salani, 2006), decremental state-space relaxation (Boland et al., 2006; Righini and Salani, 2008), heuristic pricing as long as negative RC columns can be found (Desaulniers et al., 2008), and adding buckets of multiple task-disjoint columns to the RMP in each pricing step (Desaulniers et al., 2002).

3.4.2. Branching

Our branching strategies serve two purposes: first we branch to reach integrality, and second we branch to enforce the previously relaxed synchronization constraints. To reach integrality, we use five branching strategies applied in the following sequence:

1. branching on the number of therapists (see §3.4.2),
2. branching on the types of therapists to perform a treatment (§3.4.2),
3. branching on possible service locations of treatments (§3.4.2),

4. branching on aggregated arc-sets (§3.4.2), and
5. branching on arcs (§3.4.2).

While branching strategies (1) and (5) are commonly used for VRPs, we introduce strategies (2), (3), and (4) to break fractional flows on a higher levels than in (5). When integrality is reach and if the precedence or location capacity constraint are violated, we branch on start time windows to forbid a reoccurrence of the violations. Branching are applied in the following arbitrarily chosen order:

6. branching on time windows to ensure precedence (§3.4.2),
7. branching on time windows to ensure location capacity (§3.4.2).

The search tree is explored using a best-first strategy. We are always applying strategies (1) to (5) to reach integrality, however instead of applying strategies (6) and (7) to reach feasibility with respect to the location capacity and precedence relation constraints, we could alternatively use combinatorial cuts as well (see §3.4.4). In the following, we will describe the seven branching strategies in detail.

Branching on the number of therapists.

Desirable properties of branching are: (a) a balanced branching tree, i.e. that the remaining problems in each branch are approximately equally complex, and (b) branching strategies are used such that the size of the tree does not grow excessively. For routing problems, branching on the number of vehicles (therapists) can often have a significant positive influence on the tree size (Costa et al., 2019). Let $f_{s,q}$ be the fractional number of therapists of type (s, q) in the optimal solution at a branching node, then two branches are generated: (1) forcing $\gamma_{s,q} \leq \lfloor f_{s,q} \rfloor$ and (2) forcing $\gamma_{s,q} \geq \lceil f_{s,q} \rceil$. E.g., if $\gamma_{s,q} = 1.667$, then the two branches are: (1) $\gamma_{s,q} \leq 1$ and (2) $\gamma_{s,q} \geq 2$. This branching is implemented through updating the lower and upper bounds of $\gamma_{s,q}$ instead of adding cuts to the RMP. If several therapist types (s, q) with fractional flows exist, we branch first on the type of therapist (s, q) with most fractional flow as calculated in Equation (3.15).

$$\operatorname*{arg\,min}_{(s,q)\in\mathcal{S}\times\mathcal{Q}} = |0.5 - (f_{s,q} \bmod 1)| \tag{3.15}$$

Note that branching on the number of vehicles alone is not sufficient to generate integer solutions, instead addition branching strategies are needed, e.g. arc-branching as used in Desrochers et al. (1992).

Branching on the types of therapists.

If a treatment is serviced by therapists of different types in an optimal solution of the current linear relaxation, we branch on the set of therapists, which can perform the service. Let $\mathcal{S}_i \times \mathcal{Q}_i$ be the set of therapist types that can perform treatment i. If $|\mathcal{S}_i \times \mathcal{Q}_i| \geq 2$, this set can be indexed and split in half. For both subsets, we generate a branch. To balance the branching tree, we split $\mathcal{S}_i \times \mathcal{Q}_i$ such that both halves account roughly for a flow of 0.5. We define the point μ_i, at which we split the set of therapist types as the last element that remains in subset 1 for treatment i. This split point μ_i is determined by using the formula (3.16), with f_k being the cumulated flow of therapist type k covering treatment i.

$$\mu_i = \underset{n=\{1,\ldots,|\mathcal{S}_i \times \mathcal{Q}_i|\}}{\arg\min} \left| 0.5 - \sum_{k=1}^{n} f_k \right| \tag{3.16}$$

As each PP corresponds to a specific therapist type, we can impose the branching decisions by simply removing nodes from graph G that can no longer be covered by the specific therapist type.

Branching on the locations.

When $|\mathcal{L}_i| > 1$ for a treatment i, it can happen that, in an optimal solution, treatment i is serviced by an integer number of therapists of the same type, but in more than one location. In this case, we branch on the set of possible service locations. Equivalently to §3.4.2, we split $\mathcal{L}_i$ such that the flow in the remaining halves is as close as possible to 0.5. The split position μ_i, i.e. the last location that is part of location subset 1, is determined by the following formula:

$$\mu_i = \underset{n=\{1,\ldots,|\mathcal{L}_i|\}}{\arg\min} \left| 0.5 - \sum_{k=1}^{n} f_k \right| \tag{3.17}$$

Branching on arcs flows.

Desrosiers et al. (1984) introduced branching on arc flows for the VRPTW and it has become the most popular branching strategy ever since (Costa et al., 2019).

The idea is that an arc $a \in \mathcal{A}$ with fractional flow f_a in the optimal solution at a branching node, must be part of the solution in one branch ($f_a = 1$), and must not be part of the solution in the other branch ($f_a = 0$). Compared with standard VRPTWs, the graph G in our PP contains more edges as multiple nodes exist per treatment (cf. §3.4.1). Thus, if we would apply pure arc branching, we would need more branching steps than traditional VPRTWs require. Therefore, we first branch on aggregated arc-sets, and once the flows on the aggregated arcs are integral, we continue by branching on arcs $a \in \mathcal{A}$ as defined for graph G.

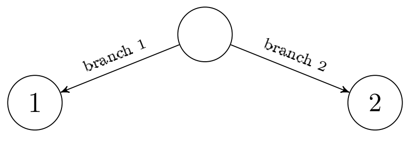
Branching on aggregated arc-sets. Let v_i be an aggregation set of all nodes belonging to the same treatment i, i.e. $v_i = \bigcup_{l \in \mathcal{L}_i} v_{i,l}$, and let (v_i, v_j) be an aggregated arc connecting two nodes v_i and v_j. Branch 1 enforcing that aggregated arc (v_i, v_j) cannot belong to the solution, is constructed by removing all arcs $(v_{i,l}, v_{j,r}) \in \mathcal{A} : l \in \mathcal{L}_i, r \in \mathcal{L}_j$ connecting any node in v_i to any node in v_j, and deleting all columns $r \in \Omega'$ containing any of these arcs. Branch 2 enforcing that arc (v_i, v_j) is used in the solution is constructed by removing all arcs $(v_{i'}, v_j) \in \mathcal{A}$ such that $i' \neq i$ and $i' \neq i_0$, and all arcs $(v_i, v_{j'}) \in \mathcal{A}$ such that $j' \neq j$ and $j' \neq i_{n+1}$ from the PPs with i_0 and i_{n+1} being the dummy treatments for leaving and entering the depot.

Branching on arcs of the PP. If the flow of all aggregated arcs is integer, however still fractional flows exist on the original arcs $\mathcal{A}$ of the PP, we branch on those as well. Branch 1 enforcing that arc $(v_{i,l}, v_{j,r})$ cannot belong to the solution is constructed by removing arc $(v_{i,l}, v_{j,r})$ from the PPs. Branch 2 enforcing that arc $(v_{i,l}, v_{j,r})$ is used in the solution is constructed by removing all arcs $(v_{i',l'}, v_{j,r}) \in \mathcal{A}$ such that $(i' \neq i \land i' \neq i_0) \lor l' \neq l$, and by removing all arcs $(v_{i,l}, v_{j',r'}) \in \mathcal{A}$ such that $(j' \neq j \land j' \neq i_{n+1}) \lor r' \neq r$.

Branching on time windows.

Once an integer feasible solution is reached, we check if this solution is also feasible for the precedence constraints and location capacity constraints. If the solution is infeasible for the original problem, we branch on the start time windows of the treatments such that the same violation If the solution is infeasible, we branch on the start time windows of the treatments such that the same violation is prevented from reoccurring. Note that violations involving the same precedence relation or capacity violations of the same location can still occur after branching on time windows, however only at a different time. Branching on start time windows belongs to branching on resource windows and has been applied in the context of vehicle routing e.g., in Desrosiers et al. (1986) and Gélinas et al. (1995).

Branching on time windows to ensure precedence. A precedence relation $(i, j) \in \mathcal{P}$ ensures that treatment i must have been finished before treatment j can start, i.e. $T_i + s_i + t_{l_i,l_j} \leq T_j$. As our algorithm works on nodes $v_{i,l}$ representing a combination of treatment i and location l, we consider start time $T_{i,l}$ associated with the node. Therefore, a precedence relation violation occurs if $T_{i,l} + s_i + t_{l,r} > T_{j,r}$ with $l \in \mathcal{L}_i$ and $r \in \mathcal{L}_j$. In that case, we branch on the start time windows of nodes $v_{i,l}$ and $v_{j,r}$. In branch 1, the start time window of node $v_{i,l}$ is adjusted such that service must start slightly earlier, and in branch 2, the start time window of node $v_{i,l}$ is adjusted such that service must start slightly later. Since i and j are part of a precedence relation, j must also start later in branch 2, i.e. the time window of node $v_{j,r}$ is adjusted as well. Figure 11 illustrates the resulting branches and the updates of the time windows TW associated with the nodes. Note that



$$TW_{i,l} = [a_{i,l}, \lceil T_{i,l} - 1 \rceil] \qquad TW_{i,l} = [\lceil T_{i,l} \rceil, b_{i,l}]$$
$$TW_{j,r} = [\lfloor T_{j,r} + 1 \rfloor, b_{j,r}]$$

Figure 11 Example: resulting branches for branching on time windows to ensure precedence relations.

we discretize time by ceiling/flooring the start times $T_{i,l}$ observed in the optimal solutions, otherwise the branching would not be guaranteed to terminate.

If a column contains a node $v_{i,l} \in \mathcal{V}$ and the start time of that node is outside of the updated time window $TW_{i,l}$, then the column is deleted from Ω'. If the time windows can no longer be adjusted, because $a_{i,l} = b_{i,l}$ for any node $v_{i,l} \in \mathcal{V}$, then node $v_{i,l}$ is removed from the PP and all columns containing $v_{i,l}$ are removed from Ω'. However, treatment i can still be served in the remaining locations $r \in \mathcal{L}_i \setminus \{l\}$. If no further location exists for i, infeasibility was proven and the current branching node can be pruned. If more than one precedence violation exists, we first branch on the largest violation in terms of time units as given in (3.18).

$$\underset{\substack{v_{i,l}, v_{j,r} \in \mathcal{V}: \\ (i,j) \in \mathcal{P}}}{\arg\max} \left(T_{i,l} + s_i + t_{l_i,l_j} - T_{j,r} \right) \tag{3.18}$$

Branching on time windows to ensure location capacity. A location capacity Q_l is violated if more than Q_l patients are scheduled to receive treatment in parallel at location l. An example of capacity violations is displayed in Figure 12. The upper part displays the time periods during which the treatments are in execution, and the lower part displays the cumulated capacity usage. Since all treatments are assigned to location l, we omit index l and write T_i, a_i and b_i instead of $T_{i,l}, a_{i,l}$ and $b_{i,l}$, etc. Let $\mathcal{T}$ be the set of start times T_i for all treatments i in an integer feasible solution, then $q_l(\mathcal{T}, t) = \{\forall i \in \mathcal{I}_l \,|\, T_i \leq t \leq T_i + s_i\}$ is a function returning the set of treatments, which are in process at time t in location l. The capacity limit is $Q_l = 2$, and a capacity violation occurs between time points $t = 6$ and $t = 10$. These are actually two violations: the first between $t = [6, 9]$ by treatments $\{i_1, i_3, i_4\}$, and the second between $t = [9, 10]$ by treatments $\{i_1, i_2, i_3\}$.

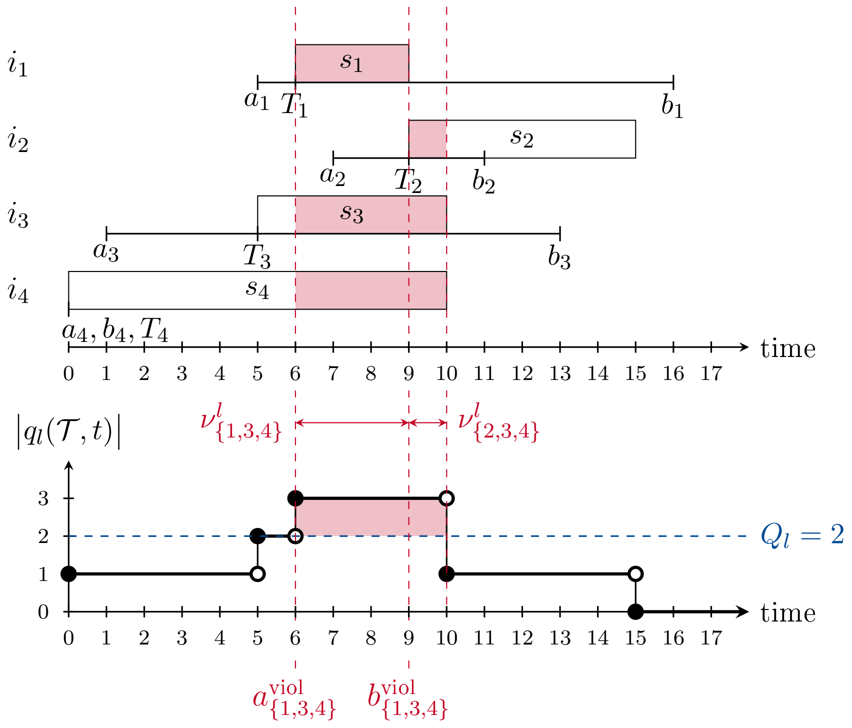
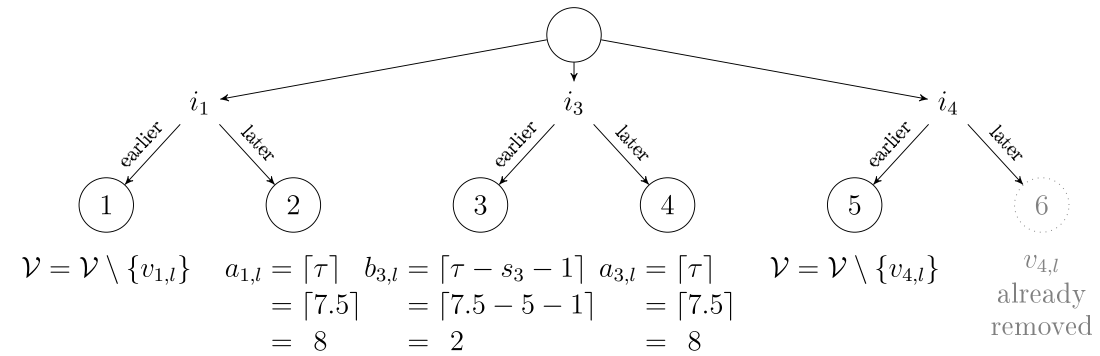


Figure 12 Example: location capacity violation.

At a branching node, we only branch on one violation at a time and choose first the violation with the longest period of resource excess. We define an arbitrary time point τ, at which we split the violation. Without loss of generalization, we assume for the example that the violation is split in half, i.e. τ is defined as $\tau = (a^{\text{viol}} + b^{\text{viol}})/2$ with a^{viol} and b^{viol} being the bounds of the violation. We generate two branches for each treatment i involved in the violation and adjust the start time windows of the treatment-location nodes: (1) such that the treatment starts either early enough that service is done before τ, i.e. $b^{\text{new}}_i = \lceil \tau - s_i - 1 \rceil$, or

(2) that service starts later than τ, i.e. $a_i^{\text{new}} = \lceil \tau \rceil$. The node is deleted from the PP, if the time windows cannot be adjusted, i.e. $(a_i + s_i > \lceil \tau - 1 \rceil) \vee (b_i < \lceil \tau \rceil)$.

The branches resulting from the first violation $\{i_1, i_3, i_4\}$ in Figure 12 are given in Figure 13. In branch 1, the first treatment i_1 cannot start so early that the violation is prevented $(a_1 + s_1 > \tau)$, therefore node $v_{1,l}$ is removed from graph G. In branch 2 however, treatment i_1 can start later than τ, and thus the earliest start time of $v_{1,l}$ is adjusted by setting $a_{1,l} = 8$. In branch 3, treatment i_3 can start so early that the violation is prevented, therefore the latest start time of $v_{3,l}$ is set $b_{3,l} = 2$. Branch 4 acts equivalently to branch 2, and branch 5 equivalently to branch 1. Branch 6 is a special case. A later start is not possible for treatment i_4 and node $v_{4,l}$ would have to be removed from the PP. However, node $v_{4,l}$ has already been removed in branch 5 and thus the resulting PPs in branch 5 and branch 6 would be identical. Therefore, branching node 6 is not generated.



i_1
i_3
i_4
earlier
later
earlier
later
earlier
later
1
2
3
4
5
6
$\mathcal{V} = \mathcal{V} \setminus \{v_{1,l}\}$
$a_{1,l} = \lceil \tau \rceil = \lceil 7.5 \rceil = 8$
$b_{3,l} = \lceil \tau - s_3 - 1 \rceil = \lceil 7.5 - 5 - 1 \rceil = 2$
$a_{3,l} = \lceil \tau \rceil = \lceil 7.5 \rceil = 8$
$\mathcal{V} = \mathcal{V} \setminus \{v_{4,l}\}$
$v_{4,l}$ already removed

Figure 13 Example: resulting branches for branching on time windows to ensure location capacity.

If a node has been deleted from the PPs, columns $r \in \Omega'$ that contain it are also removed from the RMP. Columns are removed as well if the start time of a node is outside of the updated time windows. Note that our branching strategy reduces the capacity violation by one unit at a time. Therefore, if capacity is violated by more than one unit, multiple branching steps might be needed to fully eliminate the violation.

3.4.3. MIP to correct infeasible solutions

Even if an integer feasible solution of the RMP does not fulfill the synchronization constraints (3.3) or (3.4), we can possibly transform this solution into a feasible one. The start time windows of the treatments $[a_i, b_i]$ may leave a leeway δ_i to shift start times such that the violations are eliminated. However, within each tour, the sequence of the nodes and the nodes itself must be prevailed. Otherwise, the costs of the tours could change and thus, the solution would no longer be guaranteed to be optimal for the current linear relaxation. Figure 14 displays an example of a precedence violation that can be corrected. If the start time of treatment i in tour 1 is shifted to the left and the start time of treatment j in tour 2 is shifted to the right, the violation is eliminated. The leeways to shift start times are denoted by δ_i and δ_j. Note that the leeways within a tour are interdependent, i.e. adjusting the start time of one treatment might change the leeways of the neighboring treatments as well.

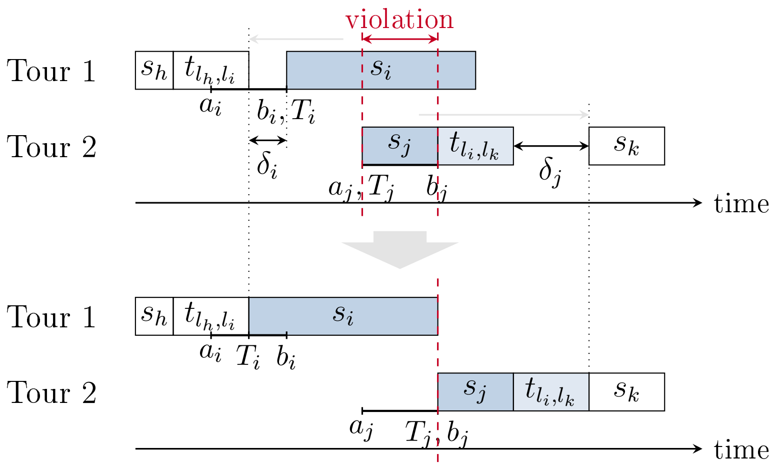


Figure 14 Example: Adjust start times to avoid precedence violation of treatment i and j.

To efficiently check if an infeasible integer solutions can be corrected, we call a MIP every time we encounter such a solution. This MIP tries to assign start times to treatments, such that the tours become feasible for the origin problem (3.1)-(3.7) including the synchronization constraints. Our MIP is based on the continuous time formulation of the RCPSP presented in Artigues et al. (2003), which decides on the sequence of activity pairs and models resources as flows through a network of activities. In the ThSRP, the treatments $i \in \mathcal{I}$ correspond to activities and the capacities of the location $l \in \mathcal{L}^{\text{bounded}}$ are the resources. One unit of the resource is consumed if a patient is treated in a capacitated location l. We distinguish between two sets of treatment pairs (i, j): (1) set $\mathcal{W}$ containing all possible pairs, and (2) set $\mathcal{U}$ for which we know that i must start before j. Let Ω^* be the set of tours used in the integer optimal solution studied and let $\mathcal{P}_r$ be the set of precedence relations within each tour r. Then $\mathcal{P}^{\Omega^*} = \bigcup_{r\in\Omega^*} \mathcal{P}_r$ is the set of all intra-tour precedence relations. The precedence relation sets can now be stated as follows:

$$\mathcal{W} = \{(i, j) \in i \in \mathcal{I} \cup \{i_0\} \times j \in \mathcal{I} \cup \{i_{n+1}\} \,|\, i \neq j\} \tag{3.19}$$

$$\mathcal{U} = \mathcal{P} \cup \mathcal{P}^{\Omega^*} \cup \left\{(i, j) \in \mathcal{W} \,|\, b_{i,l_i} \leq a_{j,l_j}\right\} \tag{3.20}$$

The MIP is formally defined as (3.21)-(3.31). Note that the optimization is stopped, once the first feasible solution is found. One feasible solution for this MIP already guarantees that the involved tours are feasible for the original problem (3.1)-(3.7).

$$\min T^{\max} \tag{3.21}$$

subject to

$$y_{i,j} = 1 \quad \forall (i,j) \in \mathcal{U} \tag{3.22}$$

$$T_{j,l_j} - (T_{i,l_i} + s_i + t_{l_i,l_j}) \geq M \cdot (y_{i,j} - 1) \quad \forall (i,j) \in \mathcal{W} \setminus \mathcal{U} \tag{3.23}$$

$$T_{j,l_j} - (T_{i,l_i} + s_i + t_{l_i,l_j}) \geq 0 \quad \forall (i,j) \in \mathcal{U} \tag{3.24}$$

$$f_{i,j,l} - y_{i,j} \leq 0 \quad \forall (i,j) \in \mathcal{W}, l \in \mathcal{L}^{\text{bounded}} \tag{3.25}$$

$$\sum_{\substack{j \in \mathcal{I} \cup \{i_{n+1}\}: \\ j \neq i}} f_{i,j,l} = r_{i,l} \quad \forall i \in \mathcal{I} \cup \{i_0\}, l \in \mathcal{L}^{\text{bounded}} \tag{3.26}$$

$$\sum_{\substack{i \in \mathcal{I} \cup \{i_0\}: \\ i \neq j}} f_{i,j,l} = r_{j,l} \quad \forall j \in \mathcal{I} \cup \{i_{n+1}\}, l \in \mathcal{L}^{\text{bounded}} \tag{3.27}$$

$$T^{\max} \geq T_{i,l_i} + s_i \quad \forall i \in \mathcal{I} \tag{3.28}$$

$$a_{i,l_i} \leq T_{i,l_i} \leq b_{i,l_i} \quad \forall i \in \mathcal{I} \tag{3.29}$$

$$f_{i,j,l} \in \{0,1\} \quad \forall (i,j) \in \mathcal{W}, l \in \mathcal{L}^{\text{bounded}} \tag{3.30}$$

$$y_{i,j} \in \{0,1\} \quad \forall (i,j) \in \mathcal{W} \tag{3.31}$$

We use binary variable $y_{i,j}$ to decide if treatment $i \in \mathcal{I}$ precedes treatment $j \in \mathcal{I}$. Binary variable $f_{i,j,l}$ decides on the flow of resource $l \in \mathcal{L}^{\text{bounded}}$ from i to j. Note that $f_{i,j,l}$ is defined as an integer variable in Artigues et al. (2003). However in the ThSRP, each treatment i has a maximum resource consumption of $r_{i,l} = 1$. Objective (3.21) minimizes the makespan $T^{\max}$, but alternative objectives would be possible as we are interested in the satisfaction of the constraints and not in the objective function value. Constraints (3.22) fix the sequences that are already implied by the selected tours. Correct separation times between preceding treatment i and succeeding treatment j are guaranteed by constraints (3.23) and (3.24). Parameter M is a sufficiently large constant that can be set to the length of the planning horizon. Constraints (3.25) connect $y_{i,j}$ and $f_{i,j,l}$, and ensure that a resource flow only exists between preceding treatments. For both, the inflow (3.26) and the outflow (3.27), the resource flow connected to a treatment must match the resource consumption $r_{i,l}$ of the treatment. The makespan $T^{\max}$ is set in (3.28), and variable domains are defined in (3.29), (3.30), and (3.31), respectively.

3.4.4. Cuts

We use three types of cuts in our BPC algorithm: limited-memory subset-row cuts to tighten the dual bound (§3.4.4), location capacity cuts to forbid infeasible assignments of treatments to locations (§3.4.4), and combinatorial Benders cuts to ensure feasibility for the synchronization constraints if branching on start time windows is disabled (§3.4.4).

Limited-Memory Subset-Row Cuts.

To obtain a better dual bound optimality cuts can be added to the RMP and subset-row cuts (SRC) introduced by Jepsen et al. (2008) have been shown to be very successful in this regard (Costa et al., 2019). We use a 3-SRC of the following form:

$$\sum_{r\in\Omega}\left\lfloor \frac{1}{2}\cdot\sum_{i\in\mathcal{C}}\sum_{l\in\mathcal{L}_i}\alpha_{i,l,r}\right\rfloor\lambda_r \leq 1 \tag{3.32}$$

with $\mathcal{C}\subseteq\mathcal{I}\wedge|\mathcal{C}|=3$. The intuition behind the 3-SRC is that for certain triples of treatments $\mathcal{C}$, we know that from the set of tours λ_r containing at least two of the treatments $i\in\mathcal{C}$ (i.e., $\sum_{i\in\mathcal{C}}\sum_{l\in\mathcal{L}_i}\alpha_{i,l,r}\geq 2$), at most one tour can be part of an integer feasible solution.

SRCs are non-robust cuts, i.e. adding these cuts changes the structure of the PP (Fukasawa et al., 2006). Specifically, an additional resource associated with each cut must be stored in the labels, which impede efficient label dominance. Pecin et al. (2017b) have introduced a limited-memory version of the SRCs (lm-SRCs), which counteracts the negative influence of the SRCs on the label dominance. Duals are only subtracted if a node memory set $\mathcal{M}$ with $\mathcal{C}\subseteq\mathcal{M}\subseteq\mathcal{I}$ is not left between visiting two nodes associated with $i,j\in\mathcal{C}$. Thus, the lm-SRCs are a weaker version of the SRCs, however, for more labels L, the resource consumption of SRCs s will be $v_s(L)=0$, and thus, leading to more dominated labels. Note that Pecin et al. (2017a) have also introduced a memory set based on arcs rather than nodes. However, we continue to use node memories as prior tests did not reveal benefits of using arc memories on our instances.

Location-Capacity Cuts.

We introduce a type of feasibility cuts, which we call location-capacity cut (LCC). For certain subsets of treatments $\mathcal{C}\subseteq\mathcal{I}$, we know, because of the start time windows, that if these treatments are assigned to the same location l, a location capacity violation is inevitable, i.e. regardless of how the start times are selected, the treatments always overlap. For a given location l and subset of treatments $\mathcal{C}$, the LCCs have the form:

$$\sum_{i \in \mathcal{C}} \beta_{i,l} \leq Q_l \tag{3.33}$$

Figure 15 provides an example of three treatments $\{i_1, i_3, i_7\}$ that will always require location capacity of three between time points 5 and 7 if serviced in location l. Note that, while SRCs are added when CG terminates, LCCs are

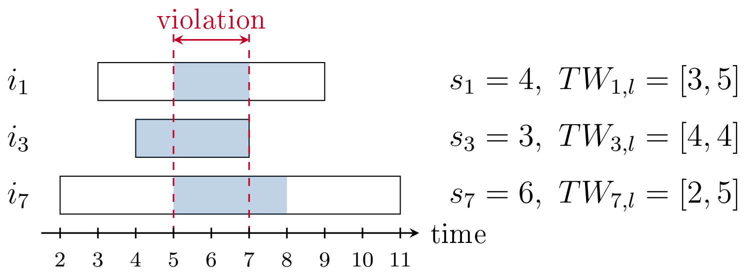

violation
i_1
i_3
i_7
$s_1 = 4,\ TW_{1,l} = [3,5]$
$s_3 = 3,\ TW_{3,l} = [4,4]$
$s_7 = 6,\ TW_{7,l} = [2,5]$
time
2
3
4
5
6
7
8
9
10
11

Figure 15 Example: Subset of treatments forming valid location-capacity cuts for location l with capacity $Q_l \in \{1, 2\}$.

added at a branching node before CG is started. The reason is that new LCCs can only be found either directly at the root node or after branching has shrunk the start time windows. As start time windows shrink, the potential overlap grows. The LCCs are robust, i.e. the cuts do not change the structure of the PPs and duals associated with these cuts are simply subtracted when a node $v_{i,l}$ is visited with i belonging to $\mathcal{C}$.

Combinatorial Benders Cuts.

In the literature, the most common approach to address violations of previously relaxed constraints is to add cuts (cf. Bruglieri et al., 2019; Froger et al., 2019). Therefore, as an alternative to branching on start time windows, we propose adding combinatorial Benders cuts to forbid the reoccurrence of solutions violating synchronization constraints.

In this branch-and-check approach, our BPC framework acts as the Benders master problem and the cut separation as the Bender subproblem. Branch-and-check is a generalization of logic-based Benders decomposition, in which the subproblem is solved whenever a feasible solution of the Benders master problem is found, i.e. in our case, a cut is generated whenever an integer feasible solution for the RMP is found that does not satisfy the synchronization constraints of the original problem. For details on branch-and-check we refer the reader to Beck (2010), for details on logic-based Benders to Hooker and Ottosson (2003), and for details on combinatorial Benders cuts to Codato and Fischetti (2004).

To exclude an infeasible integer solution s, we forbid the reuse of the exact same arc set $\mathcal{A}$ of this solution by adding the following cut:

$$\sum_{a \in \mathcal{A}_s} x_a \leq |\mathcal{A}_s| - 1 \quad \forall s \in \mathcal{S}^{\mathbb{N}} \tag{3.34}$$

where x_a is a binary variable indicating whether arc $a \in \mathcal{A}$ is used in s and $\mathcal{S}^{\mathbb{N}}$ is the set of all integer feasible solutions. Expressed in terms of the variables of model (3.8)-(3.11), the cut can be stated as follows:

$$\sum_{r \in \Omega'} \rho_{s,r} \cdot \lambda_r \leq |\mathcal{A}_s| - 1 \quad \forall s \in \mathcal{S}^{\mathbb{N}} \tag{3.35}$$

where $\rho_{s,r} \in \mathbb{N}_0$ is the number of arcs $a \in \mathcal{A}_s$ from an integer feasible solution $s \in \mathcal{S}^{\mathbb{N}}$ that column $r \in \Omega'$ contains.

We add combinatorial Benders cuts when no new SRCs can be generated. Note that the combinatorial Benders cuts are valid throughout the entire branching tree, and thus do not have to be reverted when moving up the branching tree.

3.5. Computational study

This section presents the results of our computational study and is structured as follows. In the first part §3.5.1, we detail our problem instances, and in the subsequent three parts, we investigate various aspects of our BPC algorithm. Our algorithm contains multiple components and cross-testing all possible combinations is out of scope of this work. Instead, we determined a well-working combination of components for our algorithm in prior tests, and present computational results for which each component was individually turned off to show its added value. Therefore, unless stated otherwise, our algorithm always uses branching strategies (1) to (5) to reach integrality, adds lm-SRCs and LCCs to improve the bounds, and applies the MIP checker to attempt correcting infeasible solutions. In §3.5.2, we compare the efficiency of branching on time windows and of using combinatorial Benders cuts to eliminate violations of the location capacity and precedence

Table 9 Notation used in computational study

B	Number of hospital buildings
F	Number of floors per building
gap	Optimality gap [%]
$\|\mathcal{I}\|$	Number of treatments
LB	Lower bound (dual bound)
$N^{\text{Branch}}_{\text{TW prec}}$	Number of time window branchings due to precedence violation
$N^{\text{Branch}}_{\text{TW loc}}$	Number of time window branchings due to location capacity violation
$N^{\text{Cut}}_{\text{combi}}$	Number of combinatorial cuts
$N^{\text{Cut}}_{\text{LCC}}$	Number of location capacity cuts
$N^{\text{Cut}}_{\text{SR}}$	Number of lm-SRCs
N^{infeas}	Number of instances proven to be infeasible
N^{inst}	Number of instances per hospital layout
$N^{\text{inst}}_{\text{Branch}}$	Number of instances for which time windows branching was used
N^{iter}	Number of processed branching nodes
N^{MIP}	Number of successful MIP calls to correct time window infeasible solutions
N^{non}	Number of instances, for which no feasible solution was found for the original problem
N^{opt}	Number of instances solved to optimality
time	Runtime [sec.]
UB	Upper bound (best feasible solution cost)

relation constraints. The superior approach will be part of the baseline algorithm for the subsequent tests. In §??, we evaluate the core algorithmic components of our algorithm by comparing the baseline algorithm to versions without the specific component. The evaluated components are: lm-SRCs, LCCs, and the MIP to correct infeasible solutions. In §3.5.4, we investigate the robustness of our algorithm against varying input data. Specifically, we modify the capacity limit of the locations and adjust the number of precedence relations within the instances.

The notation that we use in the computational study is summarized in Table 9. Our algorithms were coded in JAVA using Amazon Corretto 11 as JDK and executed on a Windows 10 platform employing an Intel Core i7-10510U CPU @ 2.30GHz with 16 GB of RAM. We used Gurobi 9.0.0 to solve the RMP and the checker MIP.

3.5.1. Instances

To the best of our knowledge, only Gartner et al. (2018) have studied the ThSRP. The authors worked on eight instances, three of which matched the data from a German hospital. As we needed considerably more data to reliably evaluate our algorithm and to account for different types of hospitals, we created our own instances which are available in JavaScript Object Notation (json) format at `https://www.gerad.ca/~guyd/bench-en.html`. We consider three hospital layouts varying in the number of buildings and floors in each building. The therapy center is located at one of the buildings. We consider three demand scenarios with 40, 80, and 120 treatments per day. The daily scheduling problem decomposes into a morning and evening part and the number of treatments in each part is divided by approximately two. There are three hierarchical levels of qualification. Therapists work either a regular shift or a short shift. During short shifts, therapists are only available either in the morning or the evening. For each combination of layout and demand scenario, we generated five instances, which results in $3 \cdot 3 \cdot 5 = 45$ instances in total.

3.5.2. Branching on time windows vs. combinatorial cuts

In this part of the computational study, we compare how well branching on start time windows and generating combinatorial cuts function to ensure the satisfaction of the relaxed location capacities and precedence relations. Tables 10 and 11 summarize the results for branching on time windows and imposing combinatorial cuts, respectively. Each line represents the combination of a demand scenario ($|\mathcal{I}|$) and hospital layout (combination of number of buildings B and number of floors per building F). Average values are aggregated over ten instances. Note that the ThSRP decomposes into a morning and an evening problem with approximately half the number of treatments and thus, instead of five instances, we solve ten instances. For each instance, a time limit of one hour was set, i.e. the total time limit to solve the original problem was two hours. We display the number of optimal solutions N^{opt}, the number of no feasible solution found N^{non}, we provide information about the bounds, runtime and number of iterations, cuts, and calls of the checker MIP. N^{non} is the number of runs in which we could not find feasible solutions within the given runtime limit, and this number should not be confused with the number of infeasible solutions N^{infeas}.

Branching on time windows performs better than adding combinatorial cuts. Indeed, with the former strategy, five additional instances could be solved to optimality and the average optimality gap is 0.19 lower for time window branching,

Table 10 Results for branching on start time windows

$\lvert\mathcal{I}\rvert$	B	F	N^{inst}	N^{opt}	N^{non}	LB	UB	gap	time	N^{iter}	$N^{\text{Cut}}_{\text{SR}}$	$N^{\text{Cut}}_{\text{LCC}}$	$N^{\text{Cut}}_{\text{combi}}$	N^{MIP}
40	1	6	10	10	0	57.50	57.70	0.00	119.6	1051.9	1139.6	211.0	0.0	21.4
	2	3	10	10	0	52.22	52.30	0.00	4.6	5.4	40.7	0.4	0.0	0.1
	6	1	10	10	0	50.30	50.40	0.00	9.2	18.2	86.9	3.7	0.0	0.0
80	1	6	10	10	0	85.32	85.70	0.00	116.7	129.1	409.1	0.0	0.0	0.2
	2	3	10	9	0	91.35	91.70	0.15	367.7	33.3	345.6	0.0	0.0	0.0
	6	1	10	10	0	85.22	85.30	0.00	7.4	1.4	20.0	0.0	0.0	0.1
120	1	6	10	9	1	122.20	122.78	0.00	601.4	78.0	938.6	0.0	0.0	0.0
	2	3	10	9	0	115.93	116.40	0.17	422.7	53.2	724.7	0.0	0.0	0.0
	6	1	10	10	0	116.41	116.70	0.00	124.0	10.5	172.0	0.0	0.0	0.0
Avg.				87/90	1/90	86.27	86.55	0.04	197.03	153.44	430.80	23.90	0.00	2.42

Table 11 Results for using combinatorial cuts

$\lvert\mathcal{I}\rvert$	B	F	N^{inst}	N^{opt}	N^{non}	LB	UB	gap	time	N^{iter}	$N^{\text{Cut}}_{\text{SR}}$	$N^{\text{Cut}}_{\text{LCC}}$	$N^{\text{Cut}}_{\text{combi}}$	N^{MIP}
40	1	6	10	9	0	57.33	57.80	0.49	364.0	515.6	607.4	1.6	208.0	0.1
	2	3	10	10	0	52.08	52.30	0.00	5.8	7.6	33.9	1.7	10.3	0.2
	6	1	10	8	0	50.10	50.40	0.52	721.6	374.2	435.2	94.6	393.3	0.2
80	1	6	10	9	0	85.29	85.70	0.13	387.5	167.3	345.8	0.0	89.4	0.5
	2	3	10	8	0	91.26	92.20	0.76	727.4	93.1	520.3	0.0	18.2	0.0
	6	1	10	10	0	85.23	85.30	0.00	8.7	1.7	17.9	0.0	4.8	0.1
120	1	6	10	9	1	122.20	122.78	0.00	658.8	71.8	852.2	0.0	4.9	0.0
	2	3	10	9	0	115.87	116.40	0.17	444.8	43.7	625.3	0.0	4.4	0.0
	6	1	10	10	0	116.39	116.70	0.00	178.1	10.6	173.2	0.0	4.7	0.0
Avg.				82/90	1/90	86.19	86.62	0.23	388.52	142.84	401.24	10.88	82.00	0.12

which is a relative reduction of 82.6%. Interestingly, branching on time windows required less time while performing more iterations, which we interpret as that combinatorial cuts add significant complexity to the RMP that cannot be justified by its performance. Therefore, branching on time windows will serve as the baseline algorithm in the remainder of this computational study.

3.5.3. Value of algorithmic features

We investigate the performance improvements achieved by three of our core algorithmic components: (1) limited-memory subset-row cuts (lm-SRCs), (2) location capacity cuts (LCCs), and (3) the MIP to correct infeasible solution. To measure the benefit of using the specific component, we run the baseline algorithm on all instances with and without the specific component.

The results for evaluating the lm-SRCs are given in Table 12. Without lm-SRCs, 10 instances less could be solved to optimality, no feasible solution was found for

4 additional instances, and the runtime increased by a factor of 3.62. Our results confirm the literature and emphasize that lm-SRCs should also be applied to the ThSRP.

Table 12 Value of limited-memory subset-row cuts

$\|\mathcal{I}\|$	N^{inst}	baseline					no lm-SRCs			
		N^{opt}	N^{non}	gap	time	$N^{\text{Cut}}_{\text{SR}}$	N^{opt}	N^{non}	gap	time
40	30	30	0	0.00	44.47	422.40	30	0	0.00	124.73
80	30	29	0	0.05	163.93	258.23	28	0	0.11	356.90
120	30	28	1	0.06	382.70	611.77	19	5	1.17	1660.23
Avg.		87/90	1/90	0.04	197.03	430.8	77/90	5/90	0.43	713.95
Δ							-10	+4	+0.39	+516.92

The value of adding LCCs to the RMP are shown in Table 13. The solution

Table 13 Value of location capacity cuts

$\|\mathcal{I}\|$	N^{inst}	baseline					no LCCs			
		N^{opt}	N^{non}	gap	time	$N^{\text{Cut}}_{\text{LCC}}$	N^{opt}	N^{non}	gap	time
40	30	30	0	0.00	44.47	71.7	30	0	0.00	44.00
80	30	29	0	0.05	163.93	0.0	29	0	0.05	163.33
120	30	28	1	0.06	382.70	0.0	28	1	0.06	384.43
Avg.		87/90	1/90	0.04	197.03	23.9	87/90	1/90	0.04	197.25
Δ							+0	+0	+0.00	+0.22

quality remains and the difference in runtime is insignificant. However, on average only 23.9 cuts were added per instance. In a separate run, we decreased the location capacity of the therapy centers by one unit such that the instances become more restrictive. Note that reducing the location capacity (Q_l is replaced by $Q_l - 1$) leads to higher probability of encountering infeasible solutions. For the case of reduced capacity, Table 14 shows that adding LCCs yields positive effects. Without LCCs, runtime increases by a factor of 2.28, and for 7 fewer instances infeasibility could be proved.

The value of calling the MIP to correct infeasible integer solutions, is displayed in Table 15. On our instances, an improvement could not be identified; the runtime increases by 2.06% while the solution quality remains. We performed also a test on instances with reduced location capacity, however the results were very similar; no improvement in solution quality and same runtime ($\Delta = 0.003\%$). Our explanation is that time windows are not wide enough such that shifting

Table 14 Value of location capacity cuts with reduced location capacity ($Q_l - 1$)

$\lvert\mathcal{I}\rvert$	N^{inst}	baseline ($Q_l - 1$)						no LCCs ($Q_l - 1$)				
		N^{opt}	N^{infeas}	N^{non}	gap	time	$N^{\text{Cut}}_{\text{LCC}}$	N^{opt}	N^{infeas}	N^{non}	gap	time
40	30	15	14	0	0.15	129.83	573.3	15	9	5	0.23	746.40
80	30	25	4	0	0.06	160.83	0.4	25	2	2	0.06	405.47
120	30	28	0	1	0.06	381.27	0.0	28	0	1	0.06	381.87
Avg.		68/90	18/90	1/90	0.09	223.98	191.23	68/90	11/90	8/90	0.12	511.25
Δ								+0	-7	+7	+0.03	+287.27

Table 15 Value of MIP to correct infeasible solutions

$\lvert\mathcal{I}\rvert$	N^{inst}	baseline					no MIP			
		N^{opt}	N^{non}	gap	time	N^{MIP}	N^{opt}	N^{non}	gap	time
40	30	30	0	0.00	44.47	7.17	30	0	0.00	30.77
80	30	29	0	0.05	163.93	0.10	29	0	0.05	161.83
120	30	28	1	0.06	382.70	0.00	28	1	0.06	386.30
Avg.		87/90	1/90	0.04	197.03	2.42	87/90	1/90	0.04	192.97
Δ							+0	+0	+0.00	-4.06

the start times of the treatments could resolve precedence or location capacity violation. Outpatients e.g., have a fixed start time and cannot be moved within the tours. However, adding the MIP does not slow down the optimization and can remain in the framework as it might be beneficial for others context with different data.

3.5.4. Influence of instance properties

To investigate how our algorithm behaves if input data changes, we (a) modify the capacity of the TCs, and (b) gradually reduce the number of precedence relations. To isolate the effects of modified inputs, we specifically evaluate the subsets of instances in which branching on time windows was applied in the baseline runs (§3.5.2) to eliminate violations of the synchronization constraints.

Table 16 displays the results for modified location capacities. If capacity is increased by one unit ($Q_l + 1$), the runtime, the number of branchings on time windows, and the number of generated LCCs decreased substantially compared to the baseline. All instances were solved to optimality in 0.02% of the time required to solve the baseline. Increasing the capacity limit by another unit ($Q_l + 2$) changes the results only in details. Capacity is no longer restrictive in any instance, and the runtime and solution quality remains compared to $Q_l + 1$.

However, if capacity is reduced by one unit ($Q_l - 1$), complexity increases extensively.

Table 16 Results for changed location capacity

Setup	N^{inst}	N^{infeas}	LB	UB	gap	time	N^{iter}	$N^{\text{Branch}}_{\text{TW loc}}$	$N^{\text{inst}}_{\text{Branch}}$	$N^{\text{Cut}}_{\text{LCC}}$	N^{MIP}
$Q_l - 1$	6	5	79.00	81.00	0.0247	601.83	6663.0	2836.67	1	2805.5	0.50
baseline	6	0	56.67	56.67	0.0000	213.83	1788.5	501.83	6	358.5	35.67
$Q_l + 1$	6	0	55.33	55.33	0.0000	4.50	6.5	0.17	1	0.0	0.17
$Q_l + 2$	6	0	55.33	55.33	0.0000	4.50	6.5	0.00	0	0.0	0.00

To evaluate the influence of the number of precedence relations, we generated two additional scenarios: one in which we reduce the number of precedence relations by 50% compared to the baseline, and another one, in which we remove all precedence relations. The -50%-case is generated as follows. For each instance, we iterate through the list of treatments sorted by their id and only keep every other precedence relation. The baseline has 5.46 precedence relations per instance on average, and the -50%-scenario 2.48. Note that if e.g., two instances contain each 9 precedence relations, in both instances 5 relations are removed. Therefore, in -50%-scenario slightly more than 50% of the relations are removed (54.58%). The results are displayed in Table 17. Only small differences between the baseline and the -50%-scenario can be observed for the solution quality and runtime. The reason is that in one instance[1] a precedence relation exists, which is extremely difficult to resolve. In fact, over 90% of the time to solve the 7 instances is spent for this instance and it accounts for over 97% of the time window branchings. If all precedence relations are removed, the problem becomes an easy one, which can be solved in 0.07% of the runtime compared to the baseline.

Table 17 Results for changed number of precedence relations

Setup	N^{inst}	LB	UB	gap	time	N^{iter}	$N^{\text{Branch}}_{\text{TW prec}}$	$N^{\text{inst}}_{\text{Branch}}$	N^{MIP}
baseline	7	73.43	73.43	0.0	147.00	176.71	33.57	7	0.43
precedence -50%	7	73.29	73.29	0.0	143.43	174.00	32.71	2	0.43
precedence -100%	7	73.14	73.14	0.0	10.57	3.86	0.00	0	0.00

3.6. Conclusion

We addressed the ThSP and modeled it as a variant of the VRPTW with heterogenous fleet, operations and resource synchronizations, and flexible service locations. We presented a BPC approach that solves realistic hospital instances

[1] instance i080-b1-f6-v01-Morning

in reasonable time. Branching on start time windows and combinatorial Benders cuts were used to cope with the synchronization constraints, and branching on time windows was superior. In total, we presented seven branching strategies serving different purposes and breaking structures on different levels of aggregation.

Considering location capacity is relevant for real-world applications, also outside of the hospital context. We hope that branching on time windows or other resource windows will help better solving problems involving e.g., capacitated charging and fuel stations, parking lots, or parcel delivery boxes.

4 Conclusion

This chapter summarizes our findings and provides directions for future research.

4.1. Summary

We addressed the ThSRP, a relevant problem arising in hospital operation management, and modeled it as a VRPTW-FL. The VRPTW-FL is a new variant of the VRPTW with heterogeneous fleet, flexible delivery locations, time-dependent location capacity and precedence relation, and to the best of our knowledge we are the first to address these properties in one routing problem. In therapist scheduling, routing decisions are generally not addressed in detail, however travel times account for considerable fractions of the therapists' working times, especially in larger hospitals. By addressing the ThSRP as a routing problem rather than a scheduling problem we are able to plan on a more granular level than prior works and harness the full flexibility that comes from having flexible service locations.

To solve the ThSRP we developed both, a metaheuristic and an exact solution algorithm. In Chapter 2, we proposed a hybrid ALNS framework to solve the VRPTW-FL heuristically. We started by introducing the specifics of the VRPTW-FL and discussed how our problem relates to other VRP variants. After presenting a compact mathematical formulation for the VRPTW-FL, we detailed the specific of the underlying graph structure compared to the VRPTW. The ALNS framework that we built on to solve the VRPTW-FL is enhanced with several innovative ideas, i.e. (a) a backtracking procedure in the construction phase to correct poor assignment of customers to vehicles at an early stage, (b) a guided local search (GLS) that dynamically adjusts how infeasibilities are penalized in the objective function, and (c) new neighborhoods exploiting the underlying problem structure.

Backtracking effectively improved the feasibility of the constructed solutions. For 45.5% more instances, feasible solution could be generated before starting the

ALNS. Extending the ALNS with a GLS approach helped the algorithm to traverse the solution space more efficiently. Good solutions were found already at early iterations and the quality of the solutions consistently increased throughout the entire 50′000 iterations of the ALNS. We introduced ten new destroy operators to exploit specifics of the problem structure. The results were mixed. Some of the operators performed well while other did not yield a benefit. E.g., we showed that k-means clustering is a valid alternative to clustering based on Kruskal's minimum spanning trees. On the other hand, the subroute destroy did not work at all. Operators addressing similarities between the sets of available locations for different customers performed surprisingly poorly on average. However for individual runs, the location-related operators posed a significant benefit to improve the solution quality.

After evaluating the newly introduced components, we tested the ALNS against current hospital planning as described in the literature. The ALNS outperformed the logic currently in use by improving feasibility and overall solution costs. We also evaluated different location cost functions to account for different levels of preferences to serve customers in a specific location. For our instances, a good choice seemed to be to select location costs equal to the travel times between the preferred and the assigned locations. In summary, our results suggest that our algorithm could be used in practice. The structure of the ALNS is very flexible and additional requirements could be incorporated. If the algorithm should be used for different application in which even better solution quality and faster runtimes are required, the ALNS might not be sufficient and it would be advisable to create a framework similar to the hybrid genetic algorithm as proposed by Vidal et al. (2013).

In Chapter 3 of this thesis, we developed an exact BPC algorithm and shifted the focus stronger towards the underlying healthcare problem, the ThSRP. In the literature review, we addressed VRPs with synchronizations in detail as these synchronizations are required to incorporate the location capacity and precedence constraints in the exact approach. We presented a set covering formulation of the VRPTW-FL and proposed a BPC algorithm. The most distinct component of our BPC algorithm compared to traditional exact frameworks for routing problems is that we relax the location capacity and precedence constraints in the RMP and consider these constraints later by branching on time windows. It would not be trivial to process the duals resulting from these two constraints in the pricing problems (PPs), and by relaxing these constraints we obtain PPs similar to the ones for the VRPTW with heterogeneous fleet which we can solve more easily using standard methodology. Thereby, we shift significant effort from solving the PPs to branching. Thus, branching is at the core of our algorithm and we developed seven branching strategies to enforce the previously relaxed constraints as well as additional strategies to break structures at higher levels to avoid generating unnecessary branching nodes. To tighten the formulation, we employ limited-memory subset-row cuts and propose location capacity cuts (LCCs). As an alternative to branching on start time windows to enforce the relaxed location capacity and precedence constraints, we propose combinatorial Benders cuts.

In our computational study, we showed that branching on time windows performs better than adding combinatorial cuts. All hospital instances but one were solved to optimality. Thereby, we could demonstrate the value that considering complicated constraints through branching could have. Branching on time windows has not been done in the context of vehicle routing for 25 years. However, we believe branching on resource windows, which time windows are a part of, holds the potential of elegantly solving routing problems with complicated constraints.

4.2. Future research

This dissertation should encourage other researchers, and we see numerous opportunities how this work can be extended.

Additional aspects of the underlying hospital planning problem could be incorporated, such as overtime consideration and dynamic break assignments as not all hospitals have fixed lunch breaks from 12pm-1pm. From an online operational planning perspective, quick rescheduling during the day might be needed if patients do not show up or therapists call in sick. Generally, the treatment durations of physical therapies are known. However when scheduling surgeries, considering stochastic treatment durations becomes relevant. Furthermore, fairness considerations could be incorporated, e.g. imposing that all therapists have similar workloads and travel times. Potentially, the VRPTW-FL could be extended to a multi-day planning problem in which fairness is balanced over the course of weeks or months.

From a methodological point of view, various extensions are possible as well. The ALNS was designed to be as generic as possible, however, it could also target the ThSRP more specifically. In the ThSRP, the TCs are the critical component as these are the only locations with limited capacity. The usage of the TCs could be targeted directly in the construction heuristic, but also additional neighborhoods for the ALNS could be developed.

For the BPC framework, we see three future research directions: (a) further refinement of the framework, (b) solving the VRPTW-FL differently, and (c) extending the framework so solve a broader class of routing problems.

Adjusting the time windows of multiple treatments at one branching step could reduce the number of branchings. However, one has to be very careful that the resulting subbranches still cover the entire solution space and thus optimality is guaranteed. However, covering only parts of the solution space in the subbranches could still lead to a well-performing math-heuristic. The MIP to correct infeasible solutions would be more useful if time windows were larger and more overlapping. However, if time windows get larger, more cycles can exist in the PPs and thus, the PPs should be approached as a non-elementary shortest-path problem with resource constraints (SPPRC) instead of an ESPPRC. To solve the SPPRC efficiently, ng-routes should be added to obtain partial elementary (Baldacci et al., 2011). However, incorporating ng-routes would require further changes to the existing implementation. Concerning the modeling decisions, it would be possible to leave the precedence constraints in the RMP and solve more complicated PPs with linear node costs.

The VRPTW-FL could be extended to cover additional applications, such as E-VRPs with a limited number of chargers at a charging station, or routing concrete mixers to large construction sites where synchronization between mixers and concrete pumps is required. In general, all scheduling problems including routing decisions are a promising area to apply the methods developed in this thesis.

Bibliography

Archetti, Claudia, M. Grazia Speranza, Daniele Vigo. 2014. Chapter 10: Vehicle Routing Problems with Profits. Society for Industrial and Applied Mathematics, Philadelphia, PA, 273–297.

Artigues, Christian, Philippe Michelon, Stéphane Reusser. 2003. Insertion techniques for static and dynamic resource-constrained project scheduling. European Journal of Operational Research 149(2) 249 – 267. Sequencing and Scheduling.

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, Nabila, Michel Gendreau, Jean-Yves Potvin. 2014. An adaptive large neighborhood search for a vehicle routing problem with multiple routes. Comput. Oper. Res. 41 167–173.

Balakrishnan, Anantaram, James E. Ward, Richard T. Wong. 1987. Integrated facility location and vehicle routing models: Recent work and future prospects. American Journal of Mathematical and Management Sciences 7(1-2) 35–61.

Baldacci, R., E. Bartolini, G. Laporte. 2010. Some applications of the generalized vehicle routing problem. Journal of the Operational Research Society 61(7) 1072–1077.

Baldacci, Roberto, Aristide Mingozzi, Roberto Roberti. 2011. New route relaxation and pricing strategies for the vehicle routing problem. Operations Research 59(5) 1269–1283.

Barnhart, C., E.L. Johnson, G.L. Nemhauser, M.W. Savelsbergh, P.H. Vance. 1998. Branch–and–price: Column generation for solving huge integer programs. Operations Research 46(3) 316 – 329.

Beasley, J. E., E. M. Nascimento. 1996. The vehicle routing-allocation problem: A unifying framework. TOP 4(1) 65–86.

Beaudry, Alexandre, Gilbert Laporte, Teresa Melo, Stefan Nickel. 2010. Dynamic transportation of patients in hospitals. OR Spectrum 32(1) 77–107.

Beck, J. Christopher. 2010. Checking-up on branch-and-check. David Cohen, ed., Principles and Practice of Constraint Programming – CP 2010. Springer Berlin Heidelberg, Berlin, Heidelberg, 84–98.

Bektaş, Tolga, Güneş Erdoğan, Stefan Røpke. 2011. Formulations and branch-and-cut algorithms for the generalized vehicle routing problem. Transportation Science 45(3) 299–316.

Boland, Natashia, John Dethridge, Irina Dumitrescu. 2006. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. Operations Research Letters 34(1) 58 – 68.

Brucker, Peter, Andreas Drexl, Rolf Möhring, Klaus Neumann, Erwin Pesch. 1999. Resource-constrained project scheduling: Notation, classification, models, and methods. European Journal of Operational Research 112(3) 3 – 41.

Bruglieri, M., S. Mancini, O. Pisacane. 2019. The green vehicle routing problem with capacitated alternative fuel stations. Computers & Operations Research 112 104759.

Cissé, Mohamed, Semih Yalçındağ, Yannick Kergosien, Evren Şahin, Christophe Lenté, Andrea Matta. 2017. OR problems related to home health care: A review of relevant routing and scheduling problems. Operations Research for Health Care 13-14 1 – 22.

Codato, Gianni, Matteo Fischetti. 2004. Combinatorial benders' cuts. Daniel Bienstock, George Nemhauser, eds., Integer Programming and Combinatorial Optimization. Springer Berlin Heidelberg, Berlin, Heidelberg, 178–195.

Cordeau, J-F, M. Gendreau, G. Laporte, J-Y Potvin, F. Semet. 2002. A guide to vehicle routing heuristics. Journal of the Operational Research Society 53(5) 512–522.

Cordeau, J-F, G. Laporte, A. Mercier. 2001. A unified tabu search heuristic for vehicle routing problems with time windows. Journal of the Operational Research Society 52(8) 928–936.

Costa, Luciano, Claudio Contardo, Guy Desaulniers. 2019. Exact branch-price-and-cut algorithms for vehicle routing. Transportation Science 53(4) 946–985.

De Reyck, Bert, Erik Demeulemeester, Willy Herroelen. 1999. Algorithms for scheduling projects with generalized precedence relations. Jan Weglarz, ed., Project Scheduling: Recent Models, Algorithms and Applications. Springer US, Boston, MA, 77–105.

Dees, William A., Jr., Patrick G. Karger. 1982. Automated rip-up and reroute techniques. Proceedings of the 19th Design Automation Conference. DAC '82, IEEE Press, Piscataway, NJ, USA, 432–439.

Desaulniers, G., J. Desrosiers, M.M. Solomon, eds. 2005. Column Generation. 1st ed. Springer.

Desaulniers, Guy, Jacques Desrosiers, Marius M. Solomon. 2002. Accelerating Strategies in Column Generation Methods for Vehicle Routing and Crew Scheduling Problems. Springer US, Boston, MA, 309–324.

Desaulniers, Guy, François Lessard, Ahmed Hadjar. 2008. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. Transportation Science 42(3) 387–404.

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

Desrochers, M., J. Desrosiers, M. Solomon. 1992. A new optimization algorithm for the vehicle routing problem with time windows. Operations Research 40(2) 342 – 354.

Desrochers, M., C.V. Jones, J.K. Lenstra, M.W.P. Savelsbergh, L. Stougie. 1999. Towards a model and algorithm management system for vehicle routing and scheduling problems. Decision Support Systems 25(2) 109 – 133.

Desrochers, M., J.K. Lenstra, M.W.P. Savelsbergh. 1990. A classification scheme for vehicle routing and scheduling problems. European Journal of Operational Research 46(3) 322–332.

Desrosiers, Jacques, François Soumis, Martin Desrochers. 1984. Routing with time windows by column generation. Networks 14(4) 545–565.

Desrosiers, Jacques, Francois Soumis, Martin Desrochers, Michel Sauvé. 1986. Methods for routing with time windows. European Journal of Operational Research 23(2) 236 – 245.

Drexl, Michael. 2012. Synchronization in vehicle routing–a survey of VRPs with multiple synchronization constraints. Transportation Science 46(3) 297–316.

Ebben, M.J.R., van der Heijden; M.C., A. van Harten. 2005. Dynamic transport scheduling under multiple resource constraints. European Journal of Operational Research 167 320 – 335.

Eksioglu, Burak, Arif Volkan Vural, Arnold Reisman. 2009. The vehicle routing problem: A taxonomic review. Computers & Industrial Engineering 57(4) 1472–1483.

Feillet, D., P. Dejax, M. Gendreau, C. Gueguen. 2004. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. Networks 44(3) 216 – 229.

Feillet, Dominique. 2010. A tutorial on column generation and branch-and-price for vehicle routing problems. 4OR 8(4) 407–424.

Fikar, Christian, Patrick Hirsch. 2017. Home health care routing and scheduling: A review. Computers & Operations Research 77 86 – 95.

Fink, Martin, Guy Desaulniers, Markus Frey, Ferdinand Kiermaier, Rainer Kolisch, François Soumis. 2019. Column generation for vehicle routing problems with multiple synchronization constraints. European Journal of Operational Research 272(2) 699 – 711.

Froger, Aurélien, Jorge E. Mendoza, Ola Jabali, Gilbert Laporte. 2019. The electric vehicle routing problem with capacitated charging stations. Working paper or preprint.

Fukasawa, Ricardo, Humberto Longo, Jens Lysgaard, Marcus Poggi de Aragão, Marcelo Reis, Eduardo Uchoa, Renato F. Werneck. 2006. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. Mathematical Programming 106(3) 491–411.

Gartner, Daniel, Markus Frey, Rainer Kolisch. 2018. Hospital-wide therapist scheduling and routing: Exact and heuristic methods. IISE Transactions on Healthcare Systems Engineering 8(4) 268–279.

Gélinas, Sylvie, Martin Desrochers, Jacques Desrosiers, Marius M. Solomon. 1995. A new branching strategy for time constrained routing problems with application to backhauling. Annals of Operations Research 61(1) 91–109.

Ghoniem, A., C R Scherrer, S. Solak. 2013. A specialized column generation approach for a vehicle routing problem with demand allocation. Journal of the Operational Research Society 64(1) 114–124.

Glover, Fred, Jin-Kao Hao. 2011. The case for strategic oscillation. Annals of Operations Research 183(1) 163–173.

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

Groër, Chris, Bruce Golden, Edward Wasil. 2009. The consistent vehicle routing problem. Manufacturing & Service Operations Management 11(4) 630–643.

Gunpinar, Serkan, Grisselle Centeno. 2016. An integer programming approach to the bloodmobile routing problem. Transportation Research Part E: Logistics and Transportation Review 86 94 – 115.

Hachemi, Nizar El, Michel Gendreau, Louis-Martin Rousseau. 2013. A heuristic to solve the synchronized log-truck scheduling problem. Computers & Operations Research 40(3) 666 – 673. Transport Scheduling.

Hanne, Thomas, Teresa Melo, Stefan Nickel. 2009. Bringing robustness to patient flow management through optimized patient transports in hospitals. Interfaces 39(3) 241–255.

Hans, Erwin W., Mark van Houdenhoven, Peter J. H. Hulshof. 2012. A Framework for Healthcare Planning and Control. Springer US, Boston, MA, 303–320.

Hashimoto, Hideki, Mutsunori Yagiura, Shinji Imahori, Toshihide Ibaraki. 2013. Recent progress of local search in handling the time window constraints of the vehicle routing problem. Annals of Operations Research 204(1) 171–187.

He, Qie, Stefan Irnich, Yongjia Song. 2019. Branch-and-cut-and-price for the vehicle routing problem with time windows and convex node costs. Transportation Science 53(5) 1409–1426.

Hooker, J.N., G. Ottosson. 2003. Logic-based benders decomposition. Mathematical Programming 96 33–60.

Ibaraki, T., S. Imahori, M. Kubo, T. Masuda, T. Uno, M. Yagiura. 2005. Effective local search algorithms for routing and scheduling problems with general time-window constraints. Transportation Science 39(2) 206–232.

Ioachim, Irina, Sylvie Gélinas, François Soumis, Jacques Desrosiers. 1998. A dynamic programming algorithm for the shortest path problem with time windows and linear node costs. Networks 31(3) 193–204.

Irnich, Stefan. 2008. Resource extension functions: properties, inversion, and generalization to segments. OR Spectrum 30 113–148.

Irnich, Stefan, Guy Desaulniers. 2005. Shortest Path Problems with Resource Constraints. Springer US, Boston, MA, 33–65.

Irnich, Stefan, Paolo Toth, Daniele Vigo. 2014. Chapter 1: The Family of Vehicle Routing Problems, chap. The Family of Vehicle Routing Problems. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1–33.

Jain, Anil K. 2010. Data clustering: 50 years beyond k-means. Pattern Recognition Letters 31(8) 651 – 666.

Jepsen, M., B. Petersen, S. Spoorendonk, D. Pisinger. 2008. Subset-row inequalities applied to the vehicle-routing problem with time windows. Operations Research 56 497 – 511.

Keskin, Merve, Gilbert Laporte, Bülent Çatay. 2019. Electric vehicle routing problem with time-dependent waiting times at recharging stations. Computers & Operations Research 107 77 – 94.

Kirkpatrick, S., C. D. Gelatt, M. P. Vecchi. 1983. Optimization by simulated annealing. Science 220(4598) 671–680.

Kontoravdis, G., J.F. Bard. 1995. A GRASP for the vehicle routing problem with time windows. Journal on Computing 7(1) 10–23.

Kovacs, Attila A., Sophie N. Parragh, Richard F. Hartl. 2014. A template-based adaptive large neighborhood search for the consistent vehicle routing problem. Networks 63(1) 60–81.

Kramer, Raphael, Jean-François Cordeau, Manuel Iori. 2019. Rich vehicle routing with auxiliary depots and anticipated deliveries: An application to pharmaceutical distribution. Transportation Research Part E: Logistics and Transportation Review 129 162 – 174.

Kruskal, Joseph B. 1956. On the shortest spanning subtree of a graph and the traveling salesman problem. Proceedings of the American Mathematical Society 7(1) 48–50.

Lahyani, Rahma, Mahdi Khemakhem, Frédéric Semet. 2015. Rich vehicle routing problems: From a taxonomy to a definition. European Journal of Operational Research 241(1) 1 – 14.

Lam, Edward, Pascal Van Hentenryck. 2016. A branch-and-price-and-check model for the vehicle routing problem with location congestion. Constraints 21 394 – 412.

Laporte, Gilbert. 2009. Fifty years of vehicle routing. Transportation Science 43(4) 408–416.

Laporte, Gilbert, Ibrahim H. Osman. 1995. Routing problems: A bibliography. Annals of Operations Research 61(1) 227–262.

Laporte, Gilbert, Stefan Ropke, Thibaut Vidal. 2014. Chapter 4: Heuristics for the vehicle routing problem. Society for Industrial and Applied Mathematics, Philadelphia, PA, 87–116.

Li, Yuan, Haoxun Chen, Christian Prins. 2016. Adaptive large neighborhood search for the pickup and delivery problem with time windows, profits, and reserved requests. European Journal of Operational Research 252(1) 27 – 38.

Lim, Andrew, Zhenzhen Zhang, Hu Qin. 2017. Pickup and delivery service with manpower planning in hong kong public hospitals. Transportation Science 51(2) 688–705.

Liu, Ran, Yangyi Tao, Xiaolei Xie. 2019. An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and synchronized visits. Computers & Operations Research 101 250 – 262.

Lübbecke, M. E., J. Desrosiers. 2005. Selected topics in column generation. Operations Research 53(6) 1007 – 1023.

Mancini, Simona. 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(3) 261–270.

Masson, Renaud, Fabien Lehuéd é, Olivier Péton. 2013. An adaptive large neighborhood search for the pickup and delivery problem with transfers. Transportation Science 47(3) 344–355.

Min, Hokey, Vaidyanathan Jayaraman, Rajesh Srivastava. 1998. Combined location-routing problems: A synthesis and future research directions. European Journal of Operational Research 108(1) 1 – 15.

Mitrović-Minić, Snežana, Gilbert Laporte. 2006. The pickup and delivery problem with time windows and transshipment. INFOR: Information Systems and Operational Research 44(3) 217–227.

Moccia, L., J-F Cordeau, G. Laporte. 2012. An incremental tabu search heuristic for the generalized vehicle routing problem with time windows. Journal of the Operational Research Society 63(2) 232–244.

Nagy, Gábor, Saïd Salhi. 2007. Location-routing: Issues, models and methods. European Journal of Operational Research 177(2) 649 – 672.

Nemati, Sepehr, Oleg V. Shylo, Oleg A. Prokopyev, Andrew J. Schaefer. 2016. The surgical patient routing problem: A central planner approach. INFORMS Journal on Computing 28(4) 657–673.

Ozbaygin, Gizem, Oya Ekin Karasan, Martin Savelsbergh, Hande Yaman. 2017. A branch-and-price algorithm for the vehicle routing problem with roaming delivery locations. Transportation Research Part B: Methodological 100 115 – 137.

Parragh, Sophie N., Jean-François Cordeau. 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.

Pecin, Diego, Claudio Contardo, Guy Desaulniers, Eduardo Uchoa. 2017a. New enhancements for the exact solution of the vehicle routing problem with time windows. INFORMS Journal on Computing 29(3) 489–502.

Pecin, Diego, Artur Pessoa, Marcus Poggi, Eduardo Uchoa. 2017b. Improved branch-cut-and-price for capacitated vehicle routing. Mathematical Programming Computation 9(1) 61–100.

Pisinger, David, Stefan Ropke. 2010. Large Neighborhood Search. Springer US, Boston, MA, 399–419.

Potvin, Jean-Yves, Jean-Marc Rousseau. 1993. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. European Journal of Operational Research 66(3) 331 – 340.

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

Renaud, Jacques, Gilbert Laporte, Fayez F. Boctor. 1996. A tabu search heuristic for the multi-depot vehicle routing problem. Computers & Operations Research 23(3) 229 – 235.

Reyes, Damián, Martin Savelsbergh, Alejandro Toriello. 2017. Vehicle routing with roaming delivery locations. Transportation Research Part C: Emerging Technologies 80 71 – 91.

Righini, G., M. Salani. 2006. Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. Discrete Optimization 3 255–273.

Righini, Giovanni, Matteo Salani. 2008. New dynamic programming algorithms for the resource constrained elementary shortest path problem. Networks 51(3) 155–170.

Rix, Gregory, Louis-Martin Rousseau, Gilles Pesant. 2015. A column generation algorithm for tactical timber transportation planning. Journal of the Operational Research Society 66 278 – 287.

Ropke, Stefan, David Pisinger. 2006a. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science 40(4) 455–472.

Ropke, Stefan, David Pisinger. 2006b. A unified heuristic for a large class of vehicle routing problems with backhauls. European Journal of Operational Research 171(3) 750–775. Feature Cluster: Heuristic and Stochastic Methods in Optimization Feature Cluster: New Opportunities for Operations Research.

Savelsbergh, Martin, Tom Van Woensel. 2016. 50th anniversary invited article - city logistics: Challenges and opportunities. Transportation Science 50(2) 579–590.

Schiffer, Maximilian, Grit Walther. 2018. An adaptive large neighborhood search for the location-routing problem with intra-route facilities. Transportation Science 52(2) 331–352.

Schmid, Verena, Karl F. Doerner. 2014. Examination and operating room scheduling including optimization of intrahospital routing. Transportation Science 48(1) 59–77.

Schrimpf, Gerhard, Johannes Schneider, Hermann Stamm-Wilbrandt, Gunter Dueck. 2000. Record breaking optimization results using the ruin and recreate principle. Journal of Computational Physics 159(2) 139 – 171.

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

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

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

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

Toth, Paolo, Daniele Vigo, eds. 2002b. The vehicle routing problem, chap. An overview of vehicle routing problems. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, 1–26.

Vanderbeck, F., L.A. Wolsey. 1996. An exact algorithm for IP column generation. Operations Research Letters 19(4) 151 – 159.

Vidal, Thibaut, Teodor Gabriel Crainic, Michel Gendreau, Nadia Lahrichi, Walter Rei. 2012. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. Operations Research 60(3) 611–624.

Vidal, Thibaut, Teodor Gabriel Crainic, Michel Gendreau, Christian Prins. 2013. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. Computers & Operations Research 40(1) 475 – 489.

Vidal, Thibaut, Teodor Gabriel Crainic, Michel Gendreau, Christian Prins. 2015. Time-window relaxations in vehicle routing heuristics. Journal of Heuristics 21(3) 329–358.

Vidal, Thibaut, Gilbert Laporte, Piotr Matl. 2020. A concise guide to existing and emerging vehicle routing problem variants. European Journal of Operational Research 286(2) 401 – 416.

von Boventer, Edwin. 1961. The relationship between transportation costs and location rent in transportation problems. Journal of Regional Science 3(2) 27–40.

Voudouris, Christos, Edward Tsang. 1999. Guided local search and its application to the traveling salesman problem. European Journal of Operational Research 113(2) 469 – 499.

Watson-Gandy, CDT, PJ Dohrn. 1973. Depot location with van salesmen - a practical approach. Omega 1(3) 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(3) 311–320.

WHO. 2016. Global strategy on human resources for health: Workforce 2030. URL `https://apps.who.int/iris/bitstream/handle/10665/250368/9789241511131-eng.pdf`.

Yuan, Yuan, Diego Cattaruzza, Maxime Ogier, Frédé. 2020. A branch-and-cut algorithm for the generalized traveling salesman problem with time windows. European Journal of Operational Research 286(3) 849 – 866.

Appendix

A. Notation Chapter 2 (ALNS)

Table 18 Notation ALNS: Sets and indices

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 selecting a repair operator
ρ^-	Probability of selecting 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 $l \in \mathcal{L}$
$c_{i,l}^{\text{location}}$	Cost for serving customer $i \in \mathcal{I}$ in location $l \in \mathcal{L}$
c^{na}	Cost for not assigned customer
c^{pred}	Cost for precedence violation
$c_{l,r}^{\text{travel}}$	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

Notation ALNS: Sets and indices (continued)

$I_i^{\text{na}}(s)$	Indicator function equal to 1, if customer i is not visited in solution s
$I_i^{\text{pred}}(s)$	Indicator function equal to 1, if precedence relation of customer i is violated in solution s
$I_i^{\text{tw}}(s)$	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 that 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 for 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) $r \in \Omega^+$
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 19 Notation ALNS: 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

B. Notation Chapter 3 (BPC)

Table 20 Notation BPC: Mathematical models and cuts

$\alpha_{i,l,r}$	Parameter indicating if treatment $i \in \mathcal{I}$ serviced in location $l \in \mathcal{L}_i$ is part of tour $r \in \Omega$
$\beta_{l,t,r}$	Parameter indicating if location l is occupied at time t in tour r
$\gamma_{s,q}$	Decision variable storing the n umber of therapists with shift type $s \in \mathcal{S}$ and qualification $q \in \mathcal{Q}$
λ_r	Decision variable being 1, if tour $r \in \Omega$ is selected in the RMP, 0 otherwise
μ_i	Split position (last element in branching subset 1)
$\pi_i^{(3.9)}$	Dual value associated with the assignment constraint of the RMP
$\pi_{s,q}^{(3.10)}$	Dual value associated with the convexity constraint of the RMP
$\rho_{s,r}$	Number of arcs from a solution s, which are used in a tour r
τ	Time point, at which a time window violation is split
Ω	Set of all possible tours in the master problem
$\Omega_{s,q}$	Subset of tours in the master problem for shift type $s \in \mathcal{S}$ and qualification $q \in \mathcal{Q}$
Ω'	Set of tours currently in the RMP
Ω^*	Set of tours in the optimal solution of the RMP at a branch-and-bound node
a_i	Earliest start of treatment $i \in \mathcal{I}$
$\mathcal{A}$	Set of arcs in graph G
$\mathcal{A}_s$	Arcs in solution s, which have a positive arc flow (arcs that are traversed by therapists)
b_i	Latest start of treatment $i \in \mathcal{I}$
$c_{i,l}^{\text{location}}$	Cost of performing treatment $i \in \mathcal{I}$ in location $l \in \mathcal{L}_i$
$c_{l,r}^{\text{travel}}$	Cost of traveling between locations $l \in \mathcal{L}$ and $r \in \mathcal{L}$
c_r	Total costs of a tour $r \in \Omega$
$\bar{c}_r$	Reduced cost of tour r
$\bar{c}_{v_{i,l},v_{j,r}}$	Reduced cost of an arc $a \in \mathcal{A}$ in graph G of the PP
$\mathcal{C}$	Subset of treatments involved in a cut
$f_{i,j,l}$	Binary variable deciding on the flow of resource $l \in \mathcal{L}^{\text{bounded}}$ from i to j
f_k	Flow of vehicle type k

$f_{s,q}$	Fractional number of therapists of type (s, q) in the optimal solution
G	Graph of the PP
i_0	Dummy treatment for leaving the depot
i_{n+1}	Dummy treatment for entering the depot
$\mathcal{I}$	Set of treatments
$\mathcal{I}_c$	Subset of treatments involved in cut $c \in \mathcal{C}$
$\mathcal{I}_{s,q}$	Set of treatments that can be served in shift $s \in \mathcal{S}$ with qualifications $\geq q$
$\mathcal{K}_{s,q}$	Set of therapists with shift type $s \in \mathcal{S}$ and qualification $q \in \mathcal{Q}$
l_i	Location, in which treatment i is performed
$\mathcal{L}$	Set of locations
$\mathcal{L}_i$	Set of potential locations for treatment $i \in \mathcal{I}$
$\mathcal{L}^{\text{bounded}}$	Set of locations with bounded capacities
M	Sufficiently large integer
$\mathcal{M}$	Memory set used in lm-SRCs
$\mathcal{P}$	Set defining the precedence relations between two therapies (i, j), i.e. treatment i must be finished before treatment j can start
$\mathcal{P}_r$	Set defining the precedence relations between two treatments (i, j) within a tour r
$\mathcal{P}^{\Omega^*}$	Set containing the precedence relations $\mathcal{P}_r$ of all tours r in an optimal solution, i.e. $\mathcal{P}^{\Omega^*} = \bigcup_{r \in \Omega^*} \mathcal{P}_r$
$q_l(\mathcal{T}, t)$	Function returning the set of treatments that are in execution at time t in location l
$\mathcal{Q}$	Set of qualifications
Q_l	Capacity of location $l \in \mathcal{L}$
$r_{i,l}$	Consumption of resource l by treatment i
s_i	Duration of treatment $i \in \mathcal{I}$
$\mathcal{S}$	Set of shifts for therapists
$\mathcal{S}^{\mathbb{N}}$	Set of integer feasible solutions, not necessarily fulfilling the synchronization constraints
t_{l_1,l_2}	Travel time between locations l_1 and l_2

t_{l_i,l_j}	Travel time between locations l_i and l_j, where l_i and l_j are the locations, at which treatments i and j are performed
T_i	Start time of treatment i
$T_{i,l}$	Start time of treatment i in location l
$T_{i,l,t,r}$	Decision variable being 1, if treatment $i \in \mathcal{I}$ is done at location $l \in \mathcal{L}$ at time $t \in \mathcal{T}_i$ in tour $r \in \Omega$
TW_i	Time window in which treatment i can start
$\mathcal{T}$	Set of start times T_i for all treatments $i \in \mathcal{I}$ in integer feasible solution.
$\mathcal{T}_l^{\text{bounded}}$	Set of time intervals, in which location $l \in \mathcal{L}$ has bounded capacity
$T^{\max}$	Makespan
$\mathcal{U}$	Set of treatments pairs (i,j), for which i must start before j starts
v_0	Node in graph G representing the depot (outbound)
$v_{i,l}$	Node in graph G for performing treatment $i \in \mathcal{I}$ in location $l \in \mathcal{L}_i$
v_{n+1}	Node in graph G representing the depot (inbound)
$\mathcal{V}$	Set of nodes in graph G
$\mathcal{V}'$	Set of notes without the outbound depot node $\mathcal{V}' = \mathcal{V} \setminus \{v_0\}$
$\mathcal{W}$	Set of all possible treatments pairs (i,j)
x_a	Flow over arc $a \in \mathcal{A}$
$x_{i,j}$	Flow from treatment i to j
$y_{i,j}$	Binary decision variable being 1, if treatment $i \in \mathcal{I}$ precedes treatment $j \in \mathcal{I}$, 0 otherwise

Table 21 Notation BPC: Instances and computational study

B	Number of hospital buildings
F	Number of floors per building
gap	Optimality gap [%]
$\|\mathcal{I}\|$	Number of treatments in instance
LB	Lower bound (dual bound)
$N^{\text{Branch}}_{\text{TW prec}}$	Number time window branching because of precedence violation was applied
$N^{\text{Branch}}_{\text{TW loc}}$	Number time window branching because of location capacity violation was applied
$N^{\text{Cut}}_{\text{combi}}$	Number of combinatorial cuts
$N^{\text{Cut}}_{\text{LCC}}$	Number of location capacity cuts
$N^{\text{Cut}}_{\text{SR}}$	Number of lm-SRCs
N^{infeas}	Number of instances proven to be infeasible
N^{inst}	Number of instances per hospital layout
$N^{\text{inst}}_{\text{Branch}}$	Number of instances for which time windows branching was used
N^{iter}	Number of processed branching nodes
N^{MIP}	Number of successful MIP calls to correct time window infeasible solutions
N^{non}	Number of instances, for which no feasible solution was found for the original problem
N^{opt}	Number of instances solved to optimality.
time	Runtime [sec.]
UB	Upper bound (best feasible solution)

C. Overview ALNS paramters

Table 22 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

ALNS parameters (continued)

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