Vehicle routing problems with three-dimensional loading constraints and backhauls

Schriftliche Promotionsleistung

zur Erlangung des akademischen Grades

Doctor rerum politicarum

vorgelegt und angenommen

an der Fakultät für Wirtschaftswissenschaft

Otto-von-Guericke-Universität Magdeburg

Verfasserin: Henriette Koch, M.Sc.

Geburtsdatum und -ort: 19.03.1991, Sangerhausen

Arbeit eingereicht am: 13.04.2018

Gutachter der schriftlichen Promotionsleistung:

apl. Prof. Dr. Andreas Bortfeldt

Prof. Dr. Jan Fabian Ehmke

Datum der Disputation: 13.06.2018

Contents

List of figures

List of tables

List of algorithms

List of abbreviations

1D	One-dimensional
2D	Two-dimensional
2L-CVRP	Capacitated vehicle routing problem with two-dimensional loading constraints
2L-VRP	Vehicle routing problem with two-dimensional loading constraints
2L-VRPCB	2L-VRP with clustered backhauls
2L-VRPMB	2L-VRP with mixed backhauls
2L-VRPSDP	2L-VRP with simultaneous delivery and pickup
3D	Three-dimensional
3D-SLOPP	Three-dimensional single large object placement problem
3D-SKP	Three-dimensional single knapsack problem
3L-CVRP	Capacitated vehicle routing problem with three-dimensional loading constraints
3L-PDP	Pickup and delivery problem with three-dimensional loading constraints
3L-VRP	Vehicle routing problem with three-dimensional loading constraints
3L-VRPBTW	3L-VRP with backhauls and time windows
3L-VRPCB	3L-VRP with clustered backhauls
3L-VRPCBTW	3L-VRP with clustered backhauls and time windows
3L-VRPCB(TW)	3L-VRPCB and 3L-VRPCBTW
3L-VRPDDP	3L-VRP with divisible delivery and pickup
3L-VRPDDPTW	3L-VRP with divisible delivery and pickup and time windows
3L-VRPHF	3L-VRP with a heterogeneous fleet
3L-VRPMB	3L-VRP with mixed backhauls
3L-VRPMBTW	3L-VRP with mixed backhauls and time windows
3L-VRPMB(TW)	3L-VRPMB and 3L-VRPMBTW
3L-VRPSDP	3L-VRP with simultaneous delivery and pickup

3L-VRPSDPTW	3L-VRP with simultaneous delivery and pickup and time windows
3L-VRPSDP(TW)	3L-VRPSDP and 3L-VRPSDPTW
3L-VRPTW	3L-VRP with time windows
ACO	Ant colony optimization
ALNS	Adaptive large neighbourhood search
ALNS×DBLF	Hybrid algorithm combining ALNS and DBLF
ALNS×DBLF^{SL}	Hybrid algorithm combining ALNS and DBLF^{SL}
ALNS×LS_DBLF	Hybrid algorithm combining ALNS and LS_DBLF
ALNS×LS_DBLF^{SL}	Hybrid algorithm combining ALNS and LS_DBLF^{SL}
ALNS×LS_OS	Hybrid algorithm combining ALNS and LS_OS
B&B	Branch-and-bound
B&P	Branch-and-price
BKS	Best known solution
BL	Bottom-left
BLF	Bottom-left-fill
C&P	Cutting and packing
CLP	Container loading problem
CVRP	Capacitated vehicle routing problem
DBLF	Deepest-bottom-left-fill
DBLF^{+}	Extension of the DBLF heuristic
DBLF-Comb	Combination of DBLF and DBLF^{+}
DBLF^{SL}	DBLF heuristic modified for the side loading approach
DU	Distance unit
GA	Genetic algorithm
GLS	Guided local search
GRASP	Greedy randomized adaptive search procedure
KPI	Key performance indicator
LNS	Large neighbourhood search
LIFO	Last in first out
LS	Local search
LS_DBLF	DBLF packing heuristic integrated into a LS framework
LS_DBLF^{SL}	DBLF^{SL} packing heuristic integrated into a LS framework
LSP	Loading space partition

LU	Length unit
MILP	Mixed integer linear programming
OPP	Orthogonal packing problem
OS	Open space
PPVRPTW	Pallet-packing vehicle routing problem with time windows
PSO	Particle swarm optimization
RL	Rear loading
RTS	Reactive tabu search
SA	Simulated annealing
Sav×DBLF	Hybrid algorithm combining savings and DBLF heuristic
Sav×LS_DBLF	Hybrid algorithm combining savings and LS_DBLF heuristic
Sav×LS_OS	Hybrid algorithm combining savings and LS_OS heuristic
SL	Side loading
TA	Touching area
TS	Tabu search
TSP	Travelling salesman problem
TTD	Total travel distance
VND	Variable neighbourhood descent
VNS	Variable neighbourhood search
VU	Volume unit
VRP	Vehicle routing problem
VRPB	Vehicle routing problem with backhauls
VRPBTW	Vehicle routing problem with backhauls and time windows
VRPCB	Vehicle routing problem with clustered backhauls
VRPCBTW	VRPCB with time windows
VRPMB	Vehicle routing problem with mixed backhauls
VRPMBTW	VRPMB with time windows
VRPPD	Vehicle routing problem with pickup and delivery
VRPSDP	Vehicle routing problem with simultaneous delivery and pickup
VRPSDPTW	VRPSDP with time windows
VRPTW	Vehicle routing problem with time windows
WU	Weight unit

List of symbols

I Indices

i	Customer index
j	Customer index
k	Item index
p	Position in a route
q	Item index
t	Stage index
u	Stage index
v	Vehicle/ tour index
ψ	Index for removal and insertion heuristics

II Instance and model-specific symbols

II.a Graph and sets

E	Set of directed edges
G	Weighted graph
N	Node set containing the depot and customers ($N_c = N \setminus \{0\}$)
N_c	Node set containing the customers
N^B	Node set of backhaul customers
N^L	Node set of linehaul customers
V	Vehicle set
A	Set of coordinates along the length axis
A'	Set of coordinates along the length axis without considering rotations
A_0	Set of coordinates along the length axis excluding $\alpha = L$
B	Set of coordinates along the width axis
B'	Set of coordinates along the width axis without considering rotations
B_0	Set of coordinates along the width axis excluding $\beta = W$
Γ	Set of coordinates along the height axis
Γ'	Set of coordinates along the height axis without considering rotations
Γ_0	Set of coordinates along the height axis excluding $\gamma = H$

II.b Customer-related symbols

c_{ij}	Cost of directed edge $(i, j) \in E$
DD_i	Due date of customer i
J_i	Set of items demanded by customer i
J_i^B	Set of backhaul items supplied by customer i
J_i^L	Set of linehaul items demanded by customer i
m_i	Number of items demanded by customer i
m_i^B	Number of backhaul items supplied by customer i
m_i^L	Number of linehaul items demanded by customer i
RT_i	Ready time of customer i
ST_i	Service time of customer i

II.c Item-related symbols

a_{ik}	Base area of item I_{ik} that is directly supported by any other item
d_{ik}	Weight of item I_{ik}
$\widehat{d}_{ik}$	Weight of item I_{ik} including the weight transmitted to it from above
f_{ik}	Fragility flag of item I_{ik}
h_{ik}	Height of item I_{ik}
I_{ik}	Item k demanded by customer i
I_i	Item i (alternative notation)[1]
l_{ik}	Length of item I_{ik}
$\widehat{l}_{ik}$	Length of the edge of a packed item I_{ik} that is parallel to the length axis of the loading space
$\tilde{l}_{ikjq}$	Length of the support area of item I_{jq} for item I_{ik}
p_{ik}	Load bearing strength of item I_{ik}
$supp_{ikjq}$	Base area of item I_{ik} that is supported by item I_{jq}
t_{ik}^0	First stage including a placement of item I_{ik} in a loading space
t_{ik}^1	Last stage including a placement of item I_{ik} in a loading space
w_{ik}	Width of item I_{ik}
$\widehat{w}_{ik}$	Length of the edge of a packed item I_{ik} that is parallel to the width axis of the loading space
$\widetilde{w}_{ikjq}$	Width of the support area of item I_{jq} for item I_{ik}

[1] For the sake of brevity, this shorted notation is occasionally used. That is, all symbols defined here for item I_{ik} can also be used with just one index for an item I_i.

A_{ik}	Subset of A referring to item I_{ik}
A'_{ik}	Subset of A' referring to item I_{ik}
α_{ik}	α-coordinate of the back-left-bottom corner of item I_{ik}
α^o_{ik}	α-coordinate of the front-right-top corner of item I_{ik}
B_{ik}	Subset of B referring to item I_{ik}
B'_{ik}	Subset of B' referring to item I_{ik}
β_{ik}	β-coordinate of the back-left-bottom corner of item I_{ik}
β^o_{ik}	β-coordinate of the front-right-top corner of item I_{ik}
Γ_{ik}	Subset of Γ referring to item I_{ik}
Γ'_{ik}	Subset of Γ' referring to item I_{ik}
γ_{ik}	γ-coordinate of the back-left-bottom corner of item I_{ik}
γ^o_{ik}	γ-coordinate of the front-right-top corner of item I_{ik}
δ_{ik}	Closest possible position of an operator when (un-)loading item I_{ik}
Π_{ik}	Placement of item I_{ik} consisting of its coordinates, orientation and stage interval of the item being loaded

II.d Vehicle-related symbols

D	Vehicle weight capacity
H	Vehicle loading space height
L	Vehicle loading space length
PP_v	Packing plan of vehicle v
n_v	Number of customers in route v
R_v	Route of vehicle v as a sorted subset of customers
W	Vehicle loading space width

II.e Other instance- and problem-related symbols

c_{max}	Maximum cost of any directed edge $(i, j) \in E$
M_1, M_2	"Big-M"
M_3, M_4	"Big-M"
n	Number of customers
n_B	Number of backhaul customers
n_L	Number of linehaul customers
s	Solution

$start_{iv}$	Time at which vehicle v begins service at customer location i
VSP	Vertical stability parameter
v_{max}	Number of available vehicles
v_{used}	Number of used vehicles
$z(s)$	Objective function value/ total routing cost of solution s
α	Coordinate value on the length axis
β	Coordinate value on the width axis
γ	Coordinate value on the height axis
λ	Maximum reach
$\sigma^{v}_{\alpha\beta\gamma}$	Binary support variable for point (α, β, γ) in vehicle v
$\sigma^{vB}_{\alpha\beta\gamma}$	Binary support variable for point (α, β, γ) in vehicle v considering only backhaul items
$\sigma^{vL}_{\alpha\beta\gamma}$	Binary support variable for point (α, β, γ) in vehicle v considering only linehaul items
$\varphi^{ikv}_{\alpha\beta\gamma}$	Binary occupation variable for item I_{ik}, point (α, β, γ) in vehicle v

II.f Decision variables

o_{ik}	Orientation decision variable of item I_{ik}
x^{tv}_{ij}	Binary routing decision variable for nodes i and j, vehicle v, stage t
$\pi^{iktv}_{\alpha\beta\gamma}$	Binary placement decision variable for item I_{ik}, stage t, vehicle v, point (α, β, γ)

III Symbols used in algorithms

III.a Packing

Sets

IS	Sorted item sequence (input for packing procedure)
J^{dsupp}	Set containing all J^{dsupp}_{ik}
J^{dsupp}_{ik}	Set of items directly supporting item I_{ik}
J^{LBS}_{ik}	Set of items to be considered when determining bearing loads based on a reference item I_{ik}
J^{supp}_{ik}	Set of items supporting item I_{ik}
K	Set of already placed items

P	Set of possible placement positions

Parameters

cs	Cache size (maximum number of columns in the cache matrix)
max_enum	Maximum number of items for which a full enumeration of an item sequence is tested for packing feasibility if necessary

Other variables

h_{min}	Shortest height of any item
l_{min}	Shortest horizontal edge (length/width) of any item
$load_{ik}^{unit}(ua)$	Load carried by unit area ua of item I_{ik}
lpu	Load per transmitted per unit area
$MaxShift$	Auxiliary variable for shifting an item in order to obtain reachability
ps	Percentage support
sp	Space (open space heuristic)
Π'_{ik}	Placement of item I_{ik} consisting of its coordinates and orientation

III.b Routing

Sets and graphs

E_{NG}	Edge set in the neighbour graph NG
MC	Set of missing customers
N_{NG}	Node set in the neighbour graph NG
NG	Neighbour graph
R'_v	Route of vehicle v as a sorted subset of customers excluding the depot
Rem	Set of customers to be removed from a solution
$savList$	Sorted list of all savings
Ψ	Set of heuristics

Parameters

$iter_{max}$	Termination criterion (number of iterations)
$iter_{impr}$	Termination criterion (number of iterations without improvement)
rct	Reaction parameter
seg	Segment length

t_{max}	Termination criterion (computation time limit)
η	Noise parameter
θ	Start temperature control parameter
κ	Cooling rate
ρ	Determinism parameter
ϕ_1, ϕ_2, ϕ_3	Score adjustment parameters
$\omega_1, \omega_2, \omega_3, \omega_4$	Shaw weights

Other variables

$arrival_i$	Arrival time at customer location i
c_{ij}^{NG}	Cost of edge $(i,j) \in E_{NG}$
c_{ij}^{*}	Normalized cost of edge $(i,j) \in E$ (in the range [0,1])
$\bar{c}_v$	Average distance of route v
$\bar{c}_{v\setminus i}$	Average distance of route v without customer i
cnt_ψ	Counter of heuristic ψ
$cost(p)$	Insertion cost for position p
$cost(ins_{ipv})$	Cost for inserting customer i into position p in route v
$cost'(p)$	Modified insertion cost for position p
cx_i	Coordinate (abscissa) of the position of customer i
cy_i	Coordinate (ordinate) of the position of customer i
d_i^L	Total weight of all linehaul items of customer i
d_i^B	Total weight of all backhaul items of customer i
d_v^L	Total weight of all linehaul items in route v
d_v^B	Total weight of all backhaul items in route v
DC_i	Number of connections of customer i with the depot
ins_{iv}	Best feasible insertion of customer i into route v
ins_{ivp}	Insertion of customer i into route v at position p
n_{mc}	Number of missing customers
n_{rem}	Number of customers to be removed in an ALNS iteration
$netvol_i$	Net demand volume of customer i
$netvol_{max}$	Maximum net demand volume of all customers
$netvol_{min}$	Minimum net demand volume of all customers
$netvol_i^*$	Normalized net demand volume of customer i (in the range [0,1])

no_{max}	Upper bound of interval for number of removed customers
no_{min}	Lower bound of interval for number of removed customers
pen_{mc}	Penalty term for missing customers
pen_{v}	Penalty term for violation of the tour number restriction
RT_i^*	Normalized ready time of customer i (in the range [0,1])
$regret_i$	Regret value of customer i
$relate(i,j)$	Relatedness measure between customers i and j
scr_ψ	Score of heuristic ψ
s_{best}	Best found solution
s_{curr}	Current solution
s_{init}	Initial solution
s_{next}	Neighbour of sol_{curr}
sav_{ij}	Savings of connecting customers i and j
$Temp$	Temperature (simulated annealing)
vno_i	Vehicle number to which customer i is assigned
vol_i^L	Total volume of all linehaul items of customer i
vol_i^B	Total volume of all backhaul items of customer i
vol_v^L	Total volume of all linehaul items in route v
vol_v^B	Total volume of all backhaul items in route v
y	Random number
$z'(s)$	Total cost of solution s including penalty terms
ε_{ij}	Binary variable: $= 1$, if customers i and j are in the same tour
Ω_ψ	Weight of heuristic ψ

Chapter 1

Introduction

The transportation of goods is a major logistical challenge that many companies face. Customers expect inexpensive and fast deliveries. In order to ensure this, supplying companies strive for organizing their transportations as efficiently as possible. In particular, they want to avoid driving over long distances and utilizing more vehicles than necessary. Inefficiencies can be caused, for example, by long empty runs, i.e. vehicles driving without having loaded any goods. In 2016, empty vehicle runs amounted to approximately 20.2 % of the total vehicle-kilometres in the European Union (eurostat, 2017). Realizing return transportations with delivery trucks, i.e. using them for picking up goods in the course of their delivery tours, can be an effective measure for reducing empty runs. Consequently, reductions in travelled distances, in the number of required vehicles, in fuel consumption and in CO_2 emission can be achieved.

Return transportation is called *backhauling* and it is particularly interesting in practice for returning empty packaging material. For example, food is delivered in cooling boxes to supermarkets or clothes are transported in cardboard boxes to the shops. Both types of boxes need to be returned to the warehouse. In the automotive sector, components are delivered by suppliers in transportation containers that are usually exchanged one-to-one with empty containers at the delivery location (Klug, 2010). Instead of collecting empty containers, components can also be exchanged by finished products in one route. Another example concerns the delivery of furniture or kitchen appliances to customers, and the collection of old or damaged goods from them (Gendreau et al., 2006). For instance, an old washing machine is picked up when a new one is delivered. The returned goods can have volumes (or weights) similar to the delivered goods and require the full utilization of the vehicle loading spaces. The pickup volume can also be considerable lower than the delivery volume, which applies, in particular, if the returned boxes are foldable.

In the scientific literature, delivery problems are generally known as vehicle routing problems (VRPs). Due to their great relevance in practice, VRPs are some of the most widely studied combinatorial optimization problems. In the basic version of the problem, the capacitated VRP (CVRP), goods must be delivered from a central

depot to a set of customers. A solution to the problem requires the assignment of customers to routes and the determination of routing sequences (e.g. Irnich et al., 2014b). Furthermore, backhaul transportations have gained increasing interest in research. General pickup and delivery problems comprise all VRPs with transportations from and to customers (Parragh et al., 2008). Such problems can either deal with the transportation of goods between the customer locations, or between the customers and a depot. The latter are denoted as VRPs with backhauls (VRPBs) and they are in the focus of this thesis. Three particular problems are considered: the VRP with clustered backhauls (VRPCB), the VRP with mixed backhauls (VRPMB) and the VRP with simultaneous delivery and pickup (VRPSDP). The VRPCB and the VRPMB are characterized by the division of customers into linehaul (delivery) and backhaul (pickup) customers. Applications can be found, for example, in the grocery sector. Groceries are delivered to supermarkets and picked up from suppliers (Irnich et al., 2014a). Within one route, all linehaul customers are visited before the backhaul customers in the case of the VRPCB. Linehaul and backhaul customers can be visited in the route in arbitrary sequence if a VRPMB is considered. The third variant is the VRPSDP. Here, it is assumed that all customers have both linehaul and backhaul demands. Each customer must be visited exactly once so that the vehicle delivering the linehaul goods simultaneously picks up the backhaul goods. The examples mentioned above (deliveries for supermarket, automotive manufacturers etc.) can constitute applications for the VRPSDP.

In classical VRP formulations, the demands for the transported goods (and, accordingly, vehicle capacities) are specified in one-dimensional units, representing, for example, their weights, volumes or quantities. However, often bulky goods are transported. In this case, the spatial dimensions of the goods must be taken into account when loading a vehicle, and utilizing the complete volume of the loading space is usually not possible. That is, considering only the volume (or weight) capacity of a vehicle in the planning process can result in routes that cannot be executed. Instead of (exclusively) taking capacity restrictions into account, packing plans are required for all vehicles. Therefore, the goods are explicitly assumed to be three-dimensional (3D) cuboid items in this thesis in order to provide a more realistic modelling of practically relevant problems. The resulting problems belong to the group of vehicle routing problems with three-dimensional loading constraints (3L-VRPs).

The CVRP with three-dimensional loading constraints (3L-CVRP) consists in delivering 3D items from a central depot to a set of customers. It was first presented by Gendreau et al. (2006). The authors introduce a set of packing constraints considering vertical stability, fragility, item orientation and loading sequence. This set is adopted by the majority of the researchers, who have tackled the problem subsequently. In this thesis, additional constraints regarding the robust stability of item stacks, the reachability and the load bearing strength of items are presented, modelled and implemented into heuristics for solving the packing problem.

Unlike the 3L-CVRP, the 3L-VRP with mixed backhauls (3L-VRPMB) and the 3L-VRP with simultaneous delivery and pickup (3L-VRPSDP) have been studied very rarely in the literature before. Those problems are particularly challenging as they require the simultaneous transport of linehaul and backhaul items. Several alternative loading approaches are suggested in the following for meeting the different transportation requirements.

Furthermore, time windows are of great importance in practice as (un-)loading operations might not be possible at any time. Therefore, time windows are also considered in this thesis and constitute one component of the handled integrated routing and packing problems.

A hybrid solution approach is proposed that allows for dealing with different variants of the 3L-VRP considering backhauls and time windows. The routing subproblem is tackled by means of an adaptive large neighbourhood search (ALNS) based on the approach presented by Ropke and Pisinger (2006a). A number of new operators and further modifications are proposed for improving the algorithm. A packing procedure is integrated into the ALNS in order to ensure the feasibility of the obtained solutions with respect to the packing subproblem. For this purpose, different (alternative) packing heuristics are available including rather simple construction heuristics and more sophisticated local search-based heuristics.

Extensive numerical experiments are conducted in order to evaluate the packing heuristics, the ALNS, the hybrid algorithm, different loading approaches and the impact and suitability of additional packing constraints.

This thesis is organized as follows:

In Chapter 2, all considered problem variants are described in detail. This includes the formulation of constraints regarding the respective routing problems as well as the definition of various packing constraints. The loading approaches applied to

each problem are presented. Moreover, the problems are formulated as mathematical optimization models.

An overview of the relevant literature is given in Chapter 3. The literature review focuses mainly on the VRP with time windows and with backhauls, loading problems and VRPs with loading constraints.

The hybrid solution approach is described in Chapter 4. First, several packing heuristics are described. In particular, the implementation of the packing constraints is outlined. Subsequently, the ALNS for solving the routing subproblem and the integration of the packing procedure into the routing procedure are presented.

Chapter 5 is devoted to the numerical experiments. Initially, the set-up of the numerical experiments and used instances are described, followed by the presentation and analysis of the results.

The thesis is summarized in Chapter 6. In addition, insights for research and practice are provided based on the previously presented results and an outlook to further research is given.

Chapter 2

Problem description and formulation

In this chapter, a detailed description of the problems regarded in this thesis is provided. The problem descriptions are based on the description introduced by Gendreau et al. (2006) for the 3L-CVRP. In addition, further practically relevant packing constraints, which have not been considered by Gendreau et al. (2006), are addressed.

Different combined routing and packing problems are studied. These problems include the 3L-VRP with time windows (3L-VRPTW), and 3L-VRPs with backhaul deliveries, particularly, the 3L-VRP with clustered backhauls (3L-VRPCB), the 3L-VRP with mixed backhauls (3L-VRPMB) and the 3L-VRP with simultaneous delivery and pickup (3L-VRPSDP). Time windows represent a component of all of these problems. Their characteristics are explained in greater detail in Chapter 2.1. Hereinafter, the regarded problems are collectively referred to as 3L-VRPs with backhauls and time windows (3L-VRPBTWs). In Chapter 2.2, mathematical optimization models for the considered variants of the 3L-VRPBTW are presented.

2.1 Problem description

In this section, the problem variants considered in this thesis are described. For this purpose, Chapter 2.1.1 provides a general description of 3L-VRPBTWs including the introduction of sets, constants and constraints regarding solutions, routes and packing plans. Based on this *template*, the specific problem variants can be described in a compact manner (Chapter 2.1.2).

2.1.1 Vehicle routing problems with three-dimensional loading constraints, backhauls and time windows

In a (general) 3L-VRPBTW, goods have to be transported between a central depot and a given set of customers under consideration of time windows. The customer demands can be linehaul (delivered from the depot to the customers) or backhaul demands (picked up from the customers and brought to the depot). They are represented as sets of three-dimensional, cuboid items that are transported between

the depot and the customers by means of a fleet of homogeneous vehicles. Each customer must be assigned to a vehicle, and a route (i.e. the visiting sequence of the customers) and a packing plan must be provided for each vehicle. The objective is to minimize the total travel distance (TTD).

The problem can be described using a graph-theoretic model where the nodes represent the locations of the customers and the depot. Directed edges[2] describe the road network between them. Thus, let $G = (N, E)$ be a weighted, directed graph consisting of the node set $N = \{0, 1, \ldots, n\}$, where node 0 represents the depot and the nodes $1, \ldots, n$ represent the n customers, and the edge set $E = \{(i, j) | i, j \in N\}$. N_c denotes the set of customers, i.e. $N_c = N \setminus \{0\}$. Furthermore, let c_{ij} be the non-negative cost corresponding to edge $(i, j) \in E$. The cost c_{ij} represents the distance between the two nodes i and j ($i, j \in N$). It is assumed that all vehicles travel with a speed of one distance unit per time unit. If time windows are considered, the costs c_{ij} refer to the travel time between two locations i and j. It is further assumed that the costs satisfy the triangle inequality.

$V = \{1, \ldots, v_{max}\}$ denotes the set of v_{max} available homogeneous vehicles that have a given weight capacity D and a three-dimensional cuboid loading space of length L, width W and height H. L, W and H are assumed to be integer. A vehicle's loading space is embedded in the first octant of a 3D Cartesian coordinate system, where length, width and height of the loading space are parallel to the α-, β- and γ-axis, respectively. Thus, the rear is the $W \times H$-plane at $(L, 0, 0)$.

Each customer $i \in N_c$ demands a set $J_i = \{1, \ldots, m_i\}$ of m_i cuboid items (boxes). An item I_{ik} ($i \in N_c, k \in J_i$) has a length l_{ik}, width w_{ik}, height h_{ik} and a weight d_{ik}. All item lengths, widths and heights are assumed to be integer. Depending on the problem variant, the customer and item sets can be subdivided into linehaul and backhaul sets (see below). A loading space and an example arrangement of items is shown in Figure 2.1.

Moreover, each location $i \in N$ can have a hard time window that has to be adhered to, i.e. an earliest possible start time of service RT_i ("ready time") and a latest possible start time DD_i ("due date"). Furthermore, ST_i represents the service time that is needed to load or unload all items at location $i \in N$. A location must not be approached after its due date. If a vehicle arrives before the ready time, it has

[2] In accordance with (for example) Diestel (2000), the term "directed edges" is used. An equivalent term employed in the literature is "arcs" (cf. e.g. Domschke et al., 2015).

to wait and cannot start service until the ready time.

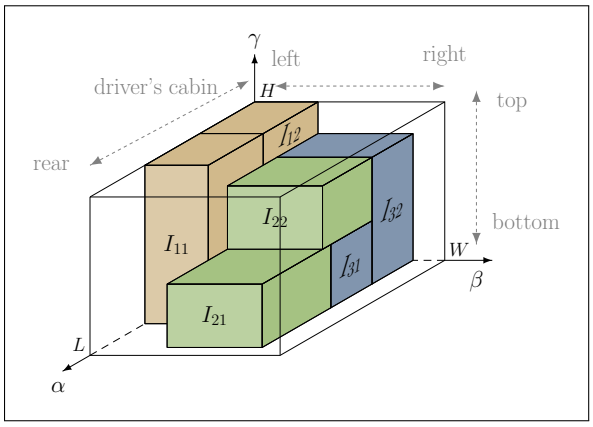


Figure 2.1: Example arrangement of items in a loading space

Let v_{used} be the number of used vehicles in a given solution consisting of v_{used} routes and v_{used} packing plans. It is assumed that each used vehicle performs exactly one route. A route R_v $(v = 1, \ldots, v_{used})$ is defined as a sorted sequence of locations $(0, i_1, \ldots, i_{n_v}, 0)$ including n_v $(n_v \geq 1)$ customer locations. Moreover, each route R_v must be provided with a packing plan PP_v that contains information about the position and orientation of each included item. Let t be the stage of a route in which a vehicle travels from one location to the next in the route. A route R_v containing n_v customer locations consists of $n_v + 1$ stages. The depot is left at the beginning of stage $t = 0$ and the vehicle returns to the depot at the end of stage $t = n_v$.
Figure 2.2 illustrates a small instance with $n = 5$, and delivery and pickup demands at each customer location. A possible solution consisting of two routes and the respective stages are shown.

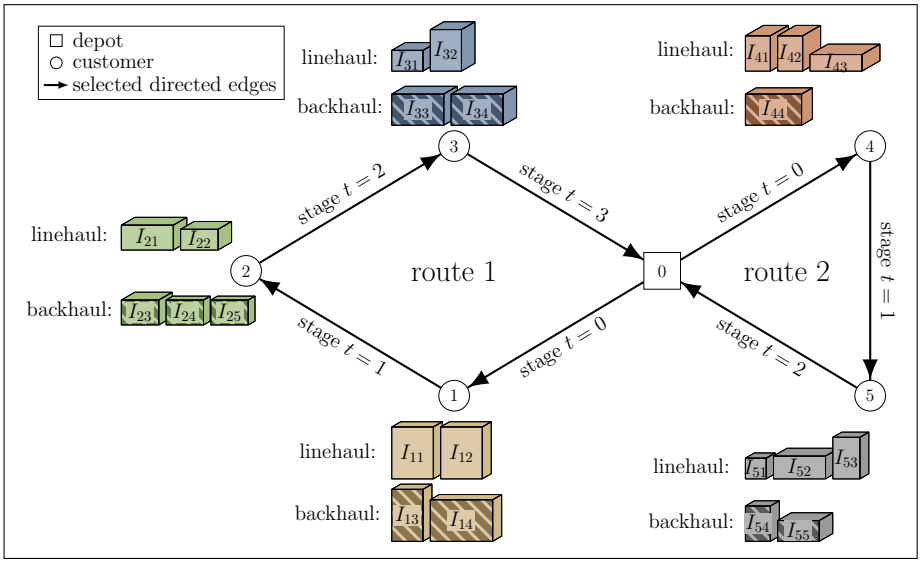


Figure 2.2: Example 3L-VRPBTW instance with possible routes

Let the point $(\alpha_{ik}, \beta_{ik}, \gamma_{ik})$ be the point where item I_{ik} $(i \in N_c, k \in J_i)$ is placed with its back-left-bottom corner.[3] This is the corner point of an item, that is closest to the origin of the coordinate system representing the loading space. $(\alpha^o_{ik}, \beta^o_{ik}, \gamma^o_{ik})$ refers to the opposite corner point, i.e. the front-right-top corner point of I_{ik}, which depends on the chosen orientation of the item. The corner points of an item with a given placement and orientation are illustrated in Figure 2.3.

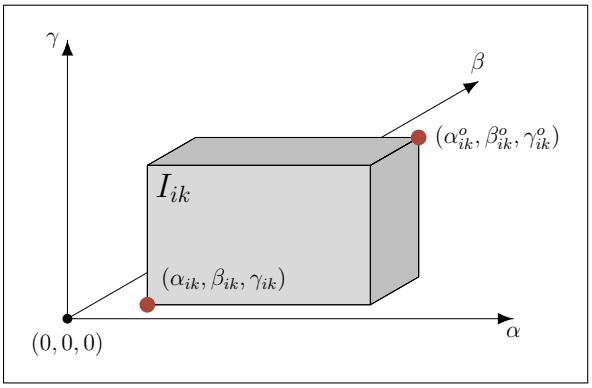


Figure 2.3: Corner points describing the placement of an item with a chosen orientation

Furthermore, let $\left[t^{start}_{ik}, t^{end}_{ik}\right]$ be the interval of stages in which item I_{ik} is within the loading space $\left(t^{start}_{ik} \leq t^{end}_{ik}\right)$. That is, $t^{start}_{ik} = 0$ if I_{ik} is delivered from the depot to customer i (linehaul). t^{end}_{ik} refers to the stage in which customer i is visited and I_{ik} is unloaded. If item I_{ik} is picked up from i and delivered to the depot (backhaul), t^{start}_{ik} is the stage in which i is left, i.e. after I_{ik} is loaded. The item remains in the loading space until $t^{end}_{ik} = n_v$ which is the stage when the vehicle returns to the depot. Two items I_{ik} and I_{jq} are simultaneously in one loading space if customers i and j are serviced in the same route (of course, also items of the same customer can be regarded), and if:

$$\max(t^{start}_{ik}, t^{start}_{jq}) \leq \min(t^{end}_{ik}, t^{end}_{jq}). \tag{2.1}$$

Consequently, a packing plan contains information about the position and orientation of a placed item as well as about the stages of a route in which the item is part of the load. The packing information of item I_{ik} can be formulated as an 8-tuple:

$$\Pi_{ik} = \left(\alpha_{ik}, \beta_{ik}, \gamma_{ik}, \alpha^o_{ik}, \beta^o_{ik}, \gamma^o_{ik}, t^{start}_{ik}, t^{end}_{ik}\right). \tag{2.2}$$

It is assumed that any reloading during the route is forbidden. Hence, each item

[3] See Figure 2.1 for the definition of "back", "left" and "bottom".

is assigned exactly one position in the corresponding packing plan. A packing plan PP_v contains information about all items transported in route R_v:

$$PP_v = \{\Pi_{ik} | i \in R_v \setminus \{0\}, k \in J_i\}. \tag{2.3}$$

An exemplary packing plan with the consecutive item arrangements in the loading space is depicted in Figure 2.4.[4] It corresponds to route 1 of the example solution from Figure 2.2. For example, the linehaul item I_{31} is loaded in the vehicle from $t_{31}^{start} = 0$ to $t_{31}^{end} = 2$. The backhaul item I_{13} is loaded from $t_{13}^{start} = 1$ to $t_{13}^{end} = 3$. Thus, both are simultaneously in the loading space from $t = \max(t_{13}^{start}, t_{31}^{start}) = 1$ to $t = \min(t_{13}^{end}, t_{31}^{end}) = 2$.

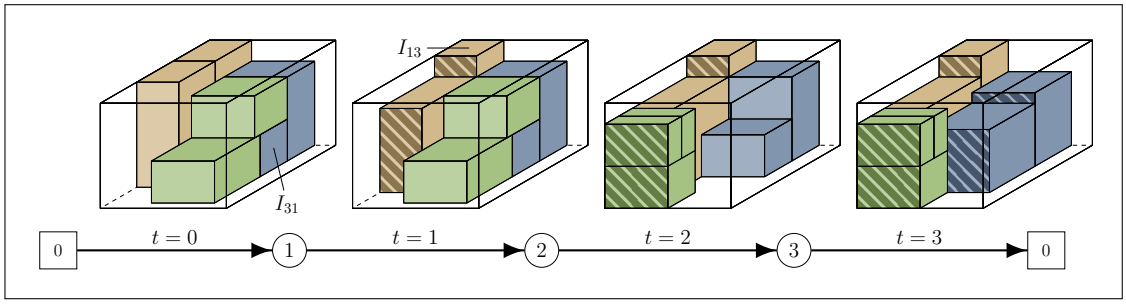


Figure 2.4: Exemplary packing plan along a route

The objective of a 3L-VRPBTW is to minimize the TTD. That is, one has to determine a feasible solution with minimum total costs of all travelled edges. Definitions of the feasibility of solutions, routes and packing plans are given below.
A solution consisting of v_{used} routes and packing plans is feasible if

(S1) all routes R_v and packing plans PP_v ($v = 1, \ldots, v_{used}$) are feasible (see below),

(S2) each packing plan PP_v contains all items of all customers i serviced in the corresponding route ($i \in R_v$), and no other,

(S3) each customer $i \in N_c$ is included in exactly one route,

(S4) the number of used vehicles does not exceed the number of available vehicles v_{max}.

[4] The depicted packing plan might not satisfy all of the packing constraints listed below.

A route R_v is feasible if

(R1) it contains at least one customer location, and starts and ends at the depot,

(R2) each customer $i \in R_v \setminus \{0\}$ is included in R_v exactly once,

(R3) the sum of weights of all simultaneously transported items does not exceed the vehicle weight capacity D,

(R4) no location $i \in R_v$ is visited after its due date DD_i (if time windows are considered).

Below, the constraints are specified more detailed and adapted to the considered problem variants if necessary (Chapter 2.1.2).
Furthermore, a feasible packing plan PP_v must contain at least one item and information about the placement for each included item. It is assumed that any reloading during the route is forbidden, i.e. each item is assigned exactly one placement position. The packing subproblem constitutes a three-dimensional orthogonal packing problem (OPP). An OPP is a satisfiability problem with the objective of determining whether or not a given set of items can be placed in a given container (loading space) under consideration of certain constraints.[5] Wäscher et al. (2007) provide a detailed typology of packing optimization problems. Leaving aside the fact that the OPP does not contain an optimization criterion, it can be classified as a three-dimensional single large object placement problem (3D-SLOPP), given that the items to be packed are weakly heterogeneous, or a three-dimensional single knapsack problem (3D-SKP), given that the items are strongly heterogeneous.
In the following, different packing constraints are presented. In this thesis, different combinations of packing constraints are considered, i.e. not necessarily all of the constraints must be satisfied. The regarded combinations are introduced below.

Geometrical constraints

The two most basic geometrical constraints (Bortfeldt and Wäscher, 2013) demand that

[5] See e.g. Fekete et al. (2007), the OPP is addressed further in the literature review (Chapter 3.2).

(P1) all items assigned to a vehicle must be positioned entirely within the vehicle's loading space, and

(P2) any two items that are simultaneously in one vehicle must not overlap.

Furthermore, it is assumed that

(P3) only orthogonal placements are permitted, i.e. the edges of the items must lie parallel to the loading space edges.

A feasible item arrangement with respect to (P1)-(P3) as well as violations of these constraints are illustrated in Figure 2.5.

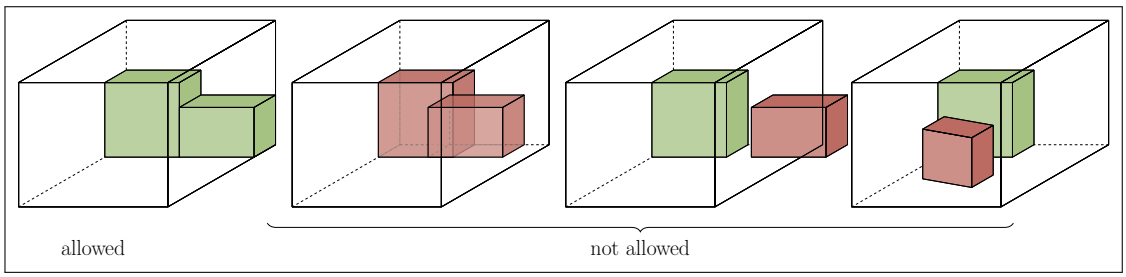


Figure 2.5: Permitted and prohibited item placements concerning the geometrical constraints

Introducing the 3L-CVRP, Gendreau et al. (2006) have applied the following four packing constraints in addition to the geometrical constraints: fixed vertical orientation, vertical stability, fragility and LIFO (loading sequence).

Fixed vertical orientation

Due to (P3), six different spatial orientations are possible for an item. Each dimension can serve as the height dimension, leading to two options for the remaining dimensions to be the length or width dimension, respectively. In practice, often "This way up!" labels can be found, for example on boxes with fragile items, and only two of the six possible orientations are permitted (see Figure 2.6).

Therefore, items have fixed vertical orientations in the packing problem considered here, i.e.

(P4) the height dimension of any item I_{ik} $(i \in N_c, k \in J_i)$ is fixed. The items can be rotated by 90° on the horizontal plane.

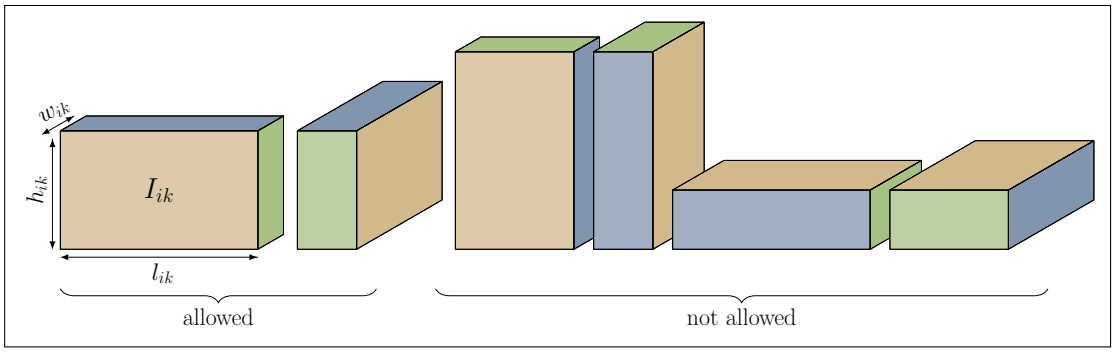


Figure 2.6: Permitted and prohibited item rotations

Vertical stability

Several aspects have to be considered when items are stacked above other items. One of these aspects is stability. A distinction can be drawn between vertical (or static) stability and horizontal (or dynamic) stability. The former refers to the capability of the items to withstand the gravity force acceleration over them, i.e. to avoid items from falling down. Dynamic stability, on the other hand, refers to moving trucks. It aims at avoiding a shift of the cargo along the length and width axes of the loading space during the transport (Junqueira et al., 2012b).

In order to achieve vertical stability,

(P5) a given threshold percentage VSP ("vertical stability parameter", $0 < VSP \leq 1$) of the bottom face of any item I_{ik} ($i \in N_c, k \in J_i$) must be supported either by the container floor or by the top faces of other items on which I_{ik} is placed.

Usually, a value $VSP \geq 0.5$ is assumed in order to guarantee stability.

The area that is supported by items placed beneath a reference item is illustrated in Figure 2.7.

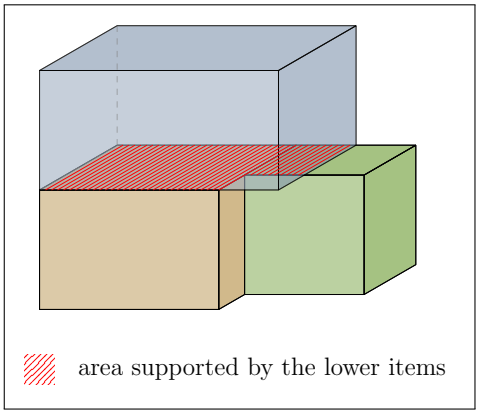


Figure 2.7: Illustration of a supported area

Horizontal stability is provided if the items are either adjacent to the side walls of the container or to other items. In the following, only vertical stability will be considered.

Fragility

A fragility flag f_{ik} is assigned to each item I_{ik} $(i \in N_c, k \in J_i)$, where

$$f_{ik} = \begin{cases} 1, & \text{if item } I_{ik} \text{ is fragile,} \\ 0, & \text{otherwise.} \end{cases} \tag{2.4}$$

The fragility constraint states that

(P6) a non-fragile item must not be placed on top of any fragile item.

As illustrated in Figure 2.8, fragile items can be stacked above each other and above non-fragile items.

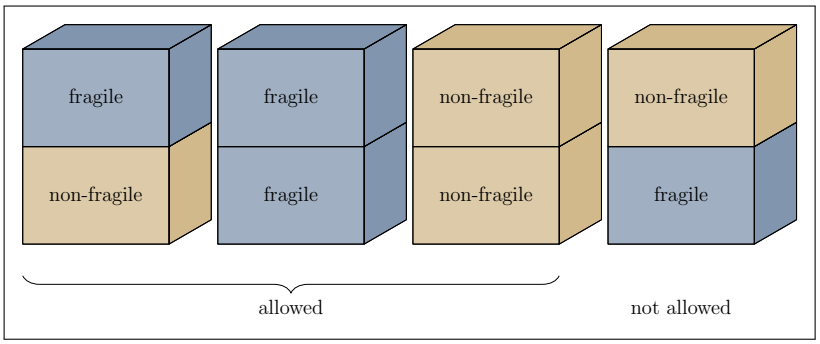


Figure 2.8: Permitted and prohibited item constellations with respect to the fragility constraint

LIFO

As mentioned above, any reloading during the route is forbidden. In particular, it must be possible to unload each item by straight movements towards the (un)loading side without moving other items. Thus, the following (general) LIFO (*last in first out*) constraint has to be satisfied:

(P7) The loading or unloading of any item must not be obstructed by any other item placed above or in front of it.

Below, the constraint is defined more detailed for the individual problem variants taking linehaul and backhaul items into account. The name LIFO refers to the loading sequence, which should represent the reverse unloading sequence.

Although it is assumed that items are loaded and unloaded by straight movements towards the unloading side (i.e. in horizontal direction), no item must be placed above another item that is delivered earlier. One reason is that the vertical stability of the item placed above should not be impaired when the item beneath is removed. Furthermore, it is assumed that items might be loaded and unloaded using forklifts. Therefore, they may need to be lifted in order to be moved (Ceschia et al., 2013). In Figure 2.9 a possible arrangement of five items is illustrated. It is assumed that they have to be delivered in the sequence (I_{11}, I_{21}, I_{31}, I_{41}, I_{51}). Thus, no item must be placed above item I_{11}, i.e. in the shaded area. As can be seen, items I_{21} and I_{31} violate this constraint. Item I_{31} must not be placed directly on top of item I_{11} because it would have to be moved when item I_{11} is unloaded. As it might be necessary to elevate an item when it is unloaded, no item may be placed anywhere above item I_{11} either (i.e. without touching its top face), like item I_{21} does.

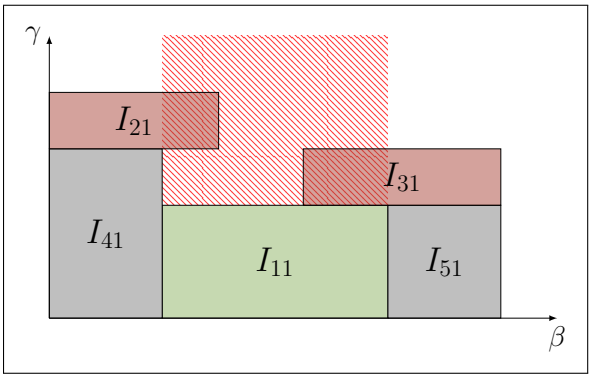


Figure 2.9: Item arrangement violating the LIFO constraint

In the following, the expression that an item I_{ik} is placed *in front of* another item I_{jq} refers to a placement of I_{ik} between item I_{jq} and the (un)loading side, which can either be the rear side or a long side (see below). Analogously, an item I_{ik} is *behind* item I_{jq} if item I_{jq} is placed between item I_{ik} and the (un)loading side. These aspects are illustrated in Figure 2.10 where the relative position of an item I_{ik} is in front of an item I_{jq} in both cases although its absolute positions are different. The (un)loading side, which is highlighted in the figure, must be considered. In Figure 2.10a, the vehicle is loaded from the rear side. In Figure 2.10b, it is loaded from a long side.

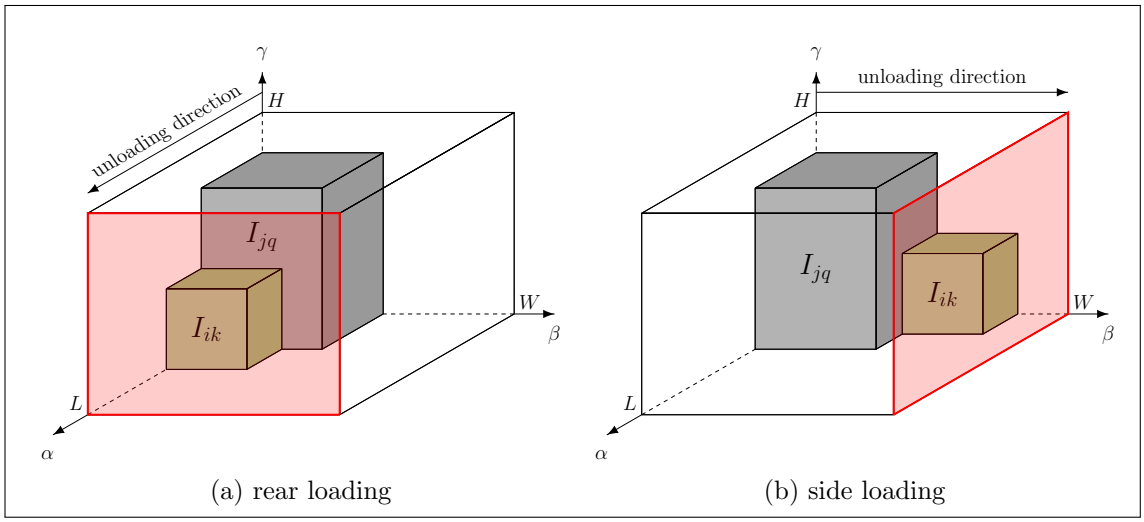


Figure 2.10: An item placed in front of another item depending on the loading approach

The above-mentioned constraints are the packing constraints introduced by Gendreau et al. (2006) for the 3L-CVRP. Some additional constraints are considered since they help to better depict real-world problems.

Reachability

This constraint considers the reachability of the items, i.e. a human operator or a forklift must be able to reach an item without having to step onto other items (Junqueira et al., 2012a; Ceschia et al., 2013):

(P8) The horizontal distance between any item I_{ik} ($i \in N_c, k \in J_i$) and the operator must not exceed a given maximum reach λ ($0 \leq \lambda \leq L$), provided that the operator stands as close as possible to the item.

λ can, for example, be the arm's length of the worker. The reachability constraint is illustrated in Figure 2.11. Suppose all items belong to different customers, who are serviced in the sequence $(1, 2, 3, 4, 5)$. When delivering item I_{11} the operator has the illustrated position. His maximum reach is illustrated by the shaded area. Thus, item I_{11} can be unloaded since it is close enough to the operator. The distance to item I_{21}, however, is too large as the operator position is restricted by the front side of item I_{31}. This arrangement would, thus, violate the reachability constraint.

In order to determine the reachability of an item I_{ik}, only those (other) items are considered that are either unloaded later than I_{ik} in the route or at the same location as it. Naturally, items that are unloaded earlier than I_{ik} do not impair the reachability of I_{ik}. With respect to other items of the same customer i, only those

items that are placed beneath I_{ik} are taken into account as they cannot be unloaded before I_{ik}. These remarks apply to linehaul items. In the case of backhauls, items picked up before the reference item are considered.

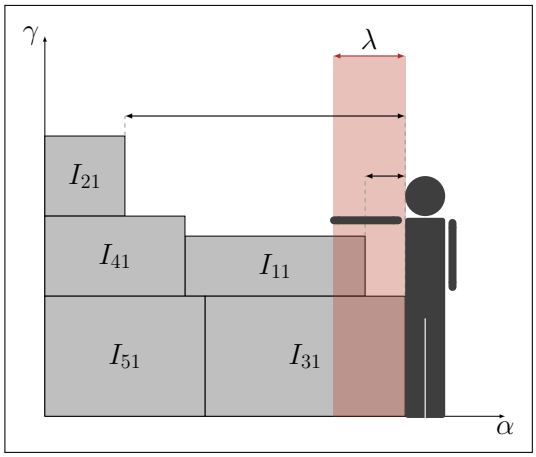
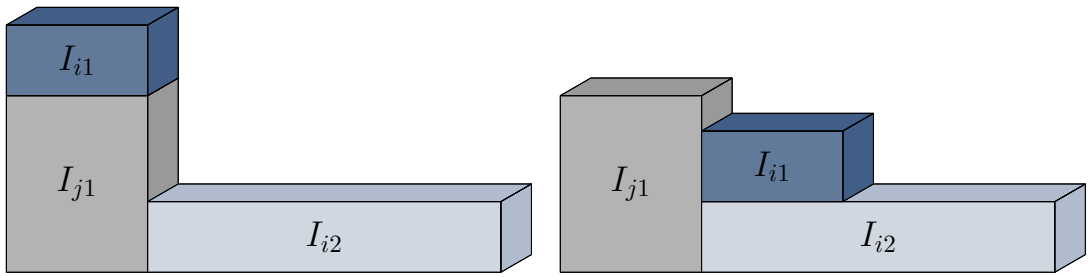


Figure 2.11: Arrangement of reachable and non-reachable items

Examples are illustrated in Figure 2.12. Items of the same customer i that can be unloaded without moving the reference item I_{i1}, such as item I_{i2} in Figure 2.12a, do not impact the reachability of item I_{i1}. They can be removed beforehand. Item I_{i2} in Figure 2.12b, which is placed below the reference item must be regarded, though, as it cannot be unloaded before item I_{i1}.



(a) Item I_{i2} does not impair the reachability of I_{i1}.

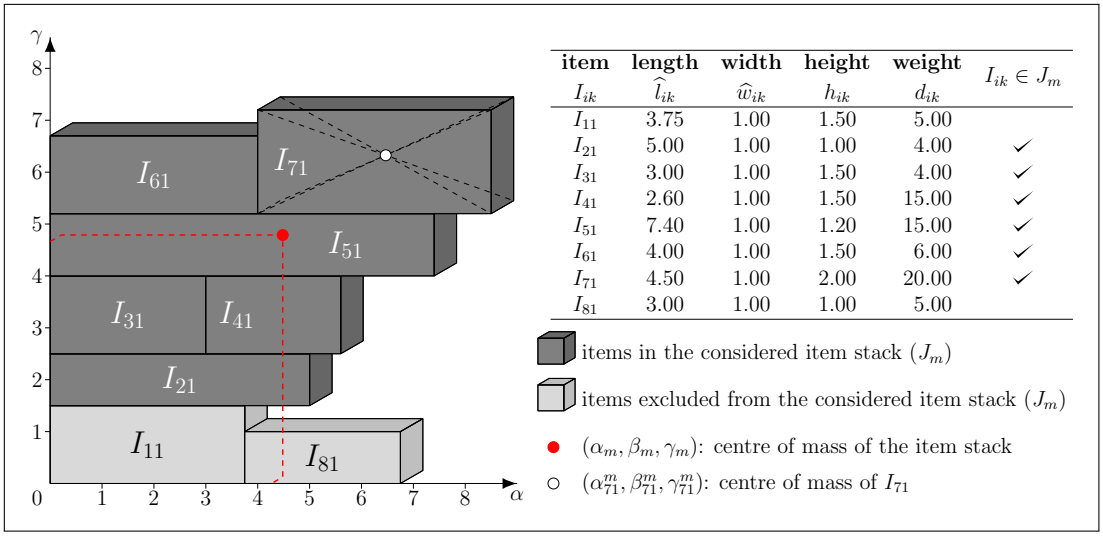
(b) Item I_{i1} cannot be reached due to I_{i2}.

Figure 2.12: Item constellations to be considered for reachability

Robust stability

The robust stability constraint is an extension of the vertical stability constraint (Ceschia et al., 2013). Although each individual item might be supported sufficiently, unstable stacks can result from considering only direct support. An example is depicted in Figure 2.13. All items are supported at least 75 % by the respective items underneath, which is sufficient to support the centres of mass of the individual

items. However, in order to be actually stable, the centre of mass of the whole stack must be supported (e.g. cf. Böge and Böge, 2017, p. 190ff.).



item I_{ik}	length $\widehat{l}_{ik}$	width $\widehat{w}_{ik}$	height h_{ik}	weight d_{ik}	$I_{ik} \in J_m$
I_{11}	3.75	1.00	1.50	5.00	
I_{21}	5.00	1.00	1.00	4.00	✓
I_{31}	3.00	1.00	1.50	4.00	✓
I_{41}	2.60	1.00	1.50	15.00	✓
I_{51}	7.40	1.00	1.20	15.00	✓
I_{61}	4.00	1.00	1.50	6.00	✓
I_{71}	4.50	1.00	2.00	20.00	✓
I_{81}	3.00	1.00	1.00	5.00	

Figure 2.13: Item stack with an unsupported centre of mass

This is not the case in the shown example. Let $(\alpha_m, \beta_m, \gamma_m)$ be the coordinates of the centre of mass of the current item stack and J_m be the set of items considered in the stack. Furthermore, $(\alpha^m_{ik}, \beta^m_{ik}, \gamma^m_{ik})$ denotes the coordinates of the centre of mass of item I_{ik}. It is assumed that all items have the same density at every point. Hence, the mass centre of an item is at its geometrical centre, i.e. where its space diagonals intersect (exemplarily illustrated for I_{71} in Figure 2.13). The centre of mass of the stack is determined as follows:

$$\alpha_m = \frac{\sum_{I_{ik} \in J_m} d_{ik} \cdot \alpha^m_{ik}}{\sum_{I_{ik} \in J_m} d_{ik}}, \tag{2.5}$$

$$\beta_m = \frac{\sum_{I_{ik} \in J_m} d_{ik} \cdot \beta^m_{ik}}{\sum_{I_{ik} \in J_m} d_{ik}}, \tag{2.6}$$

$$\gamma_m = \frac{\sum_{I_{ik} \in J_m} d_{ik} \cdot \gamma^m_{ik}}{\sum_{I_{ik} \in J_m} d_{ik}}. \tag{2.7}$$

In the example in Figure 2.13, the relevant stack consists of: $J_m = \{I_{21}, I_{31}, I_{41}, I_{51}, I_{61}, I_{71}\}$. Items I_{11} and I_{81} are not regarded because I_{11} supports the stack and I_{81} does not support any other item. The stack would tilt over the edge of I_{11} at $\alpha = 3.75$ as its centre of mass exceeds it ($\alpha_m = 4.27$).

In order to prevent such situations, the robust stability constraint is formulated as follows:

(P9) A given threshold percentage VSP of the bottom face of any item I_{ik} ($i \in$

$N_c, k \in J_i$) must be supported either by the container floor or – continuously until the container floor, i.e. at any height – by items below I_{ik} in the respective item stack.

The parameter value can also be assumed to be $VSP \geq 0.5$ in order to ensure robust stability.

If (P9) is satisfied, the centre of mass of an item stack is always supported. For the evaluation of the robust stability of a reference item, all items are considered whose parallel projections to the α-β-plane of the loading space (subsequently called α-β-projections) overlap with the α-β-projection of the reference item and that (directly or indirectly) support it.

In the following, it is to be distinguished whether an item is directly or indirectly supported by another item. An item I_{jq} supports another item I_{ik} *directly* if the bottom face of item I_{ik} touches the top face of item I_{jq}. An item I_{jq} supports another item I_{ik} *indirectly*, if it directly supports any item that directly supports item I_{ik}. Moreover, if an item I_{jq} directly supports any item that indirectly supports item I_{ik}, I_{jq} also supports I_{ik} indirectly.

In the example in Figure 2.13, item I_{71} is directly supported by item I_{51}, and indirectly supported by the items I_{11}, I_{21}, I_{31} and I_{41}. However, only the items I_{21}, I_{41} and I_{51} need to be considered for the robust stability as they are actually beneath I_{71}, i.e. their α-β-projections are overlapping. The supporting areas provided by I_{21}, I_{41} and I_{51} are illustrated in Figure 2.14.

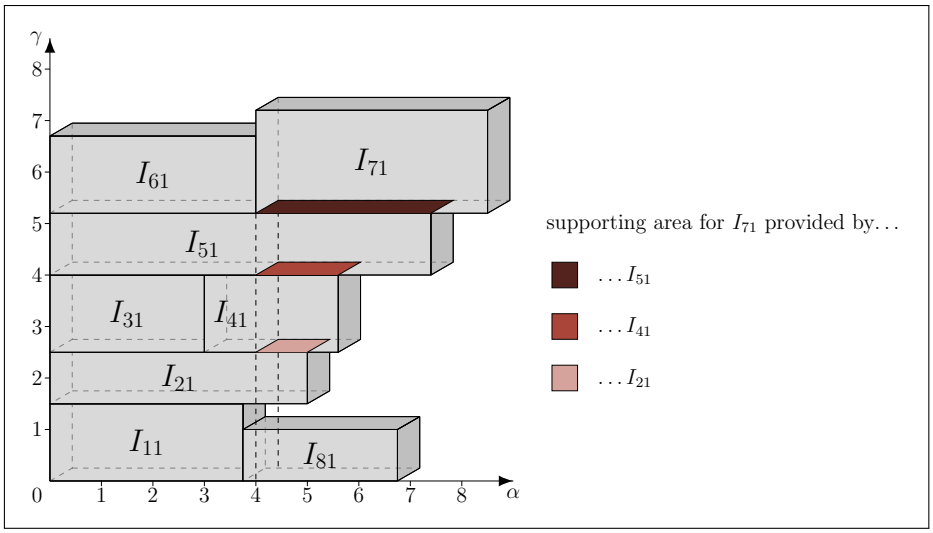


Figure 2.14: Item stack with direct and indirect support for a reference item

Assuming that $VSP = 0.75$, I_{51} offers sufficient support, whereas items I_{21} and I_{41}

do not. Hence, the robust stability constraint is violated in this example. Item I_{81} is also placed beneath item I_{71}. Yet, it does not offer any support for it and must, thus, not be considered for the robust stability of I_{71}.

Load bearing strength

If an item is stacked on top of another item, the above-mentioned fragility of these items can be considered in order to assure their integrity. Additionally (or alternatively), the load bearing strength of an item can be taken into account, i.e. the weight an item can bear per area unit without being damaged. Hereafter, let p_{ik} be the load bearing strength of item I_{ik}. Thus,

(P10) the maximal admissible pressure p_{ik} that may be applied anywhere on the top face of any item I_{ik} must not be exceeded by the items placed above item I_{ik} $(i \in N_c, k \in J_i)$.

p_{ik} is given in weight units per area unit. It is assumed that the weight is evenly distributed over the bottom area of an item and that there is no distinction between the load bearing strength of the centre and the edges of an item (Bischoff, 2006; Junqueira et al., 2012b).

The load bearing strength constraint can be modelled and implemented in different ways. A prominent assumption is that the weight of an item is transmitted (only) straight downwards (Ratcliff and Bischoff, 1998). However, considering the actual physical circumstances, the weight is rather spread and distributed over all items that support – directly or indirectly – a given item (cf. e.g. Böge and Böge, 2017, p. 406ff.). Both approaches are considered in this thesis and are explained in greater detail in Chapter 2.2.1.

This quantitative restriction can also replace the qualitative fragility restriction that was introduced earlier. In this case, instead of having a fragility flag, fragile items are assigned a load bearing strength of zero or a very low load bearing strength, respectively.

In this thesis, different sets of packing constraints are regarded. They are presented in Table 2.1. First, the set of constraints as it is utilized by Gendreau et al. (2006) is applied (C1). Furthermore, the additional constraints *reachability*, *robust stability* and *load bearing strength* are considered individually in addition to set C1 (C2-

C4). In the course of this, the robust stability (P9) replaces the vertical stability constraint (P5) and the fragility constraint (P6) is replaced by the load bearing strength constraint (P10). Finally, all of the additional constraints are taken into account, replacing former constraints if necessary (C5).

Table 2.1: Overview of the sets of packing constraints

set	**packing constraints**							
	GC (P1)-(P3)	VO (P4)	VS (P5)	FR (P6)	LIFO (P7)	R (P8)	RS (P9)	LB (P10)
C1	✓	✓	✓	✓	✓			
C2	✓	✓	✓	✓	✓	✓		
C3	✓	✓		✓	✓		✓	
C4	✓	✓	✓		✓			✓
C5	✓	✓			✓	✓	✓	✓

FR: fragility, GC: geometrical constraints, LB: load bearing strength, R: reachability, RS: robust stability, VO: vertical orientation, VS: vertical stability

2.1.2 Variants of the 3L-VRPBTW

The general problem description presented in Chapter 2.1.1 is used as a basis for describing the specific variants of the 3L-VRPBTW in a compact manner.

First, the 3L-VRP with time windows (3L-VRPTW) is presented. The VRP with time windows (VRPTW) is one of the most important and most frequently studied variants of the VRP. Furthermore, the 3L-VRPTW is a good starting point for the model as well as for the solution algorithm. In addition, problem variants with backhauls are examined. General pickup and delivery problems contain VRPs that include both delivery and pickup operations. In the past decades, different variants of them have been studied and modelled, which can be classified as depicted in Figure 2.15 (Parragh et al., 2008). On the one hand, a VRP with backhauls (VRPB) deals with the transportation of goods between a depot and the customers. On the other hand, a vehicle routing problem with pickup and delivery (VRPPD) regards the transportation of goods between customer locations, i.e. after leaving the depot, a vehicle collects the goods at pickup locations and transports them to delivery locations. While these problem types have been studied frequently in the literature regarding (one-dimensional) VRPs, there is only very little research so far for the class of 3L-VRPs (see Chapter 3).

In this thesis, the focus is on the transportation between the depot and the cus-

tomers. In particular, three variants are to be considered, namely the 3L-VRPCB, 3L-VRPMB and 3L-VRPSDP. An additional variant would be the 3L-VRP with divisible delivery and pickup (3L-VRPDDP), where customers have both linehaul and backhaul request and – as opposed to the 3L-VRPSDP – can be visited up to two times in order to conduct the delivery and pickup separately. This problem is not considered in this thesis as it is of lower relevance in practice.

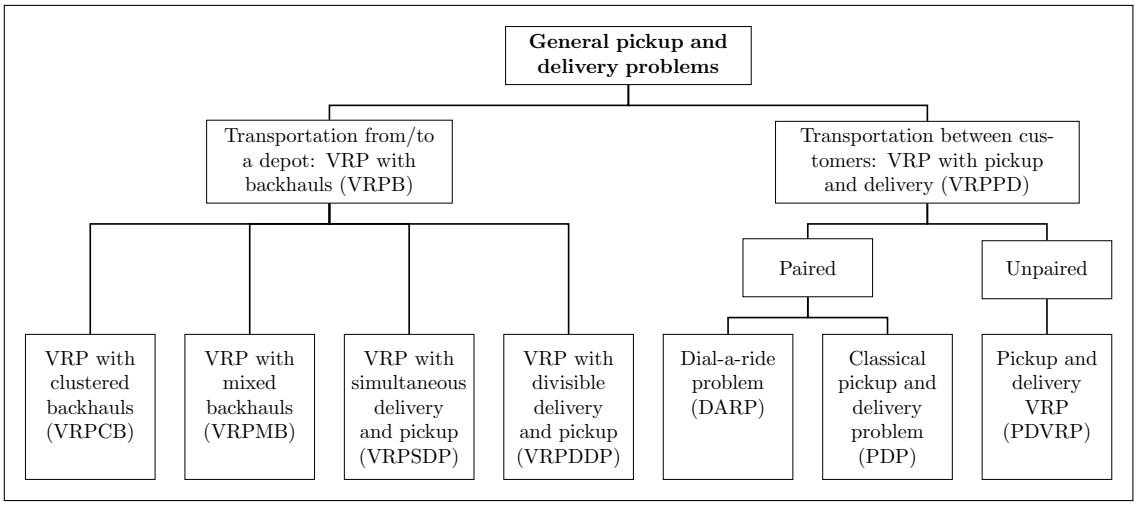


Figure 2.15: Pickup and delivery problems (Adopted from Parragh et al., 2008, p. 23)

In the following subchapters, the regarded variants are described in detail. The notations introduced in Chapter 2.1.1 are applied, too. Additional constants and sets are introduced if necessary.

In the following, the problem variants with backhauls are studied with and without time windows. The variants without time windows are denoted as 3L-VRPCB, 3L-VRPMB and 3L-VRPSDP. The variants with time windows are referred to as 3L-VRPCBTW, 3L-VRPMBTW and 3L-VRPSDPTW. For the sake of brevity, the denotations 3L-VRPCB(TW), 3L-VRPMB(TW), and 3L-VRPSDP(TW) will be utilized in order to refer to both variants with *and* without time windows. All problems are summarized in the class of the 3L-VRPBTW.

2.1.2.1 The 3L-VRP with time windows

The 3L-VRPTW is a generalization of the VRPTW, which deals with the delivery of goods to customers ensuring the arrival at the customer locations within given time windows (e.g. Moura and Oliveira, 2009). Thus, the 3L-VRPTW does not consider any backhaul transportations and consists in (exclusively) transporting three-dimensional items from the depot to the customers under consideration of time

windows.[6] Due to the lack of backhauls, the 3L-VRPTW contains a comparatively simple packing subproblem as only linehaul items have to be taken into account. The optimization criterion, the solution and routing constraints ((S1)-(S4), (R1)-(R4)) can be adopted from the general 3L-VRPBTW. The packing constraints (P1)-(P6), (P8)-(P10) are also applied unchanged to the 3L-VRPTW. The LIFO constraint (P7) is formulated more precisely as only linehaul items need to be considered:

(P7a) If customer j ($j \in N_c$) is visited later than customer i ($i \in N_c$) in the same route, no item I_{jq} ($q \in J_j$) of customer j may be placed in front of or above any item I_{ik} ($k \in J_i$) of customer i.

Two different loading approaches are taken into account for the 3L-VRPTW. They are depicted in Figure 2.16 where the arrows indicate the unloading direction.

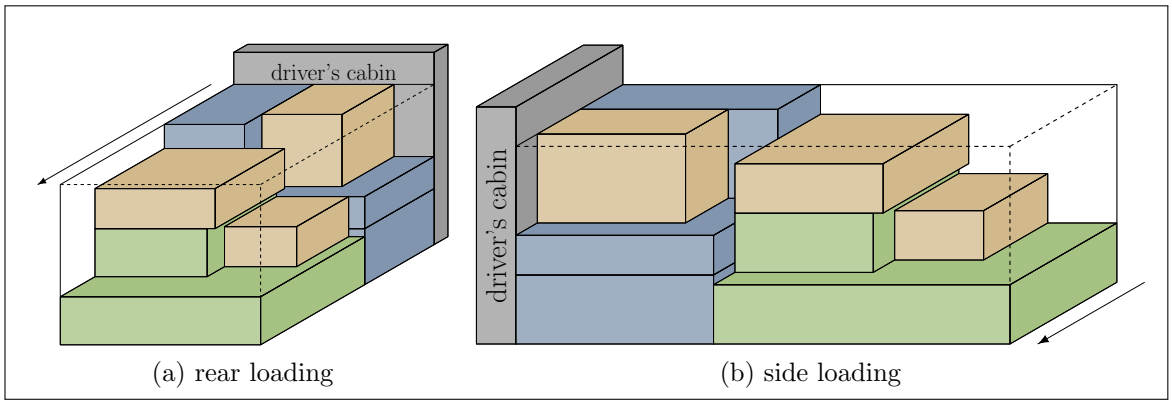


Figure 2.16: Loading approaches for the 3L-VRPTW

First, loading the vehicle from the rear is considered (Figure 2.16a). This is the standard loading approach considered in the 3L-VRP literature. Apart from rear loading vehicles, so-called tautliners are also commonly used vehicle types in practice. They are loaded from a long side which facilitates the loading and unloading processes (Figure 2.16b). This approach (in the following referred to as side loading) is applied here with the limitation of (un)loading the vehicles only from *one* long side.

The packing constraints are applied equally to the two approaches. With respect to the LIFO and reachability constraints the unloading direction must be regarded (cf. Figure 2.10).

[6] The problem variant without time windows, i.e. the 3L-CVRP, is not regarded in this thesis.

A compact problem formulation could read as follows: Three-dimensional items need to be delivered from a central depot to a set of customers using a given fleet of v_{max} homogeneous vehicles. A solution contains v_{used} tuples (R_v, PP_v). A tuple (R_v, PP_v) contains a route R_v for each used vehicle v ($v = 1, \ldots, v_{used}$) and the corresponding packing plan PP_v. In order to be feasible, a solution must fulfil (S1)-(S4) and each route must adhere to (R1)-(R4). The packing plans must satisfy the constraints (P1)-(P4) and (P7a), and the remaining relevant constraints of the respective constraint sets (cf. Table 2.1). A feasible solution is to be determined that minimizes the TTD.
The following variants are regarded: The problem is considered with time windows, with the constraint sets C1-C5 and the loading approaches rear and side loading.

2.1.2.2 The 3L-VRP with clustered backhauls and time windows

The 3L-VRP with clustered backhauls (and time windows; 3L-VRPCB(TW)) constitutes a generalization of the VRP with clustered backhauls. In this problem, each customer is either a linehaul or a backhaul customer and within a route, the linehaul customers are visited before the backhaul customers (e.g. cf. Bortfeldt et al., 2015). Thus, within a route of a solution for a 3L-VRPCB(TW), three-dimensional items have to be transported from the depot to linehaul customers. Subsequently, (other) items are picked up from backhaul customers and brought to the depot.[7] The set of customers N_c is, therefore, divided into two subsets of customers: Let n_L of all n customers be the number of linehaul customers. The set $N^L = \{1, \ldots, n_L\}$ denotes the set of linehaul customers and the set $N^B = \{n_L + 1, \ldots, n\}$ denotes the set of backhaul customers. As before, a fleet of v_{max} homogeneous vehicles is available for realizing the transportations.
The optimization criterion, the solution and routing constraints ((S1)-(S4), (R1)-(R4)) are equally valid for the 3L-VRPCB(TW). Moreover, the following constraint applies to this problem:

(R5) Within one route, all linehaul customers must be visited before the first backhaul customer can be visited.

Due to constraint (R5), linehaul and backhaul items are never transported simultaneously by the individual vehicles as all linehaul items are delivered before the first

[7] A route may also contain only linehaul customers or only backhaul customers.

backhaul item can be picked up. Thus, it is impossible that backhaul items block the unloading of linehaul items and the packing constraints can be applied separately to linehaul and backhaul items. The constraints (P1)-(P6), (P8)-(P10) are equally valid for both linehaul and backhaul items in the way they are formulated above. Moreover, (P7a) is applicable to the linehaul items in the packing plans. The following constraint is formulated for the backhaul items:

(P7b) If backhaul customer j ($j \in N^B$) is visited *earlier* than backhaul customer i ($i \in N^B$) in the same route, no item I_{jq} ($q \in J_j$) of customer j may be placed in front of (cf. Figure 2.10) or above any item I_{ik} ($k \in J_i$) of customer i.

This way, backhaul items can be loaded by straight shifts in negative α-direction (Bortfeldt et al., 2015).

As linehaul and backhaul items are transported separately, the loading approaches rear and side loading (as explained for the 3L-VRPTW) can also be applied to the 3L-VRPCB(TW).

A compact problem formulation could read as follows: Three-dimensional items need to be delivered from a central depot to a set of linehaul customers, and picked up from a set of backhaul customers and brought to the depot using a given fleet of v_{max} homogeneous vehicles. Within each route, all linehaul customers must be visited before the first backhaul customer is approached (R5). A solution contains v_{used} tuples (R_v, PP_v). A tuple (R_v, PP_v) contains a route R_v for each used vehicle v ($v = 1, \ldots, v_{used}$) and the corresponding packing plan PP_v.

In order to be feasible, a solution must fulfil (S1)-(S4) and each route must adhere to (R1)-(R3), (R5) and, optionally, (R4) (time windows). The packing plans must satisfy (P1)-(P4), the LIFO constraint and the remaining relevant constraints of the respective constraint sets (cf. Table 2.1). The LIFO constraint (P7a) is applied to linehaul items and the LIFO constraint (P7b) is applied to backhaul items. A feasible solution is to be determined that minimizes the TTD.

The following variants are regarded: The problem is considered with and without time windows, with the constraint sets C1-C5 and the loading approaches rear and side loading.

2.1.2.3 The 3L-VRP with mixed backhauls and time windows

The 3L-VRP with mixed backhauls (and time windows; 3L-VRPMB(TW)) is a generalization of the VRP with mixed backhauls (VRPMB). Similarly to the VRPCB, each customer is either a linehaul or a backhaul customer. Yet, linehaul and backhaul customers can be visited in mixed sequences (e.g. Parragh et al., 2008). Hence, the customers can also be divided into the two subsets N^L and N^B. Allowing mixed customer sequences complicates the loading problem within the 3L-VRPMB(TW), as linehaul and backhaul items can be in the vehicle at the same time. Thus, constraints, such as the weight constraint or non-overlapping constraint, must be regarded not only for linehaul and backhaul items separately. They have to be considered for every stage of a route for both linehaul and backhaul items together in order to guarantee that backhaul items loaded along the way do not violate the restrictions.

In Figure 2.17, an example instance is illustrated with four linehaul and four backhaul customers within one route. The total weight of each customer demand is given in square brackets. A VRPCB solution (Figure 2.17a) as well as a VRPMB solution (Figure 2.17b) is depicted for this instance. Let the weight capacity of the vehicle be 30 weight units.

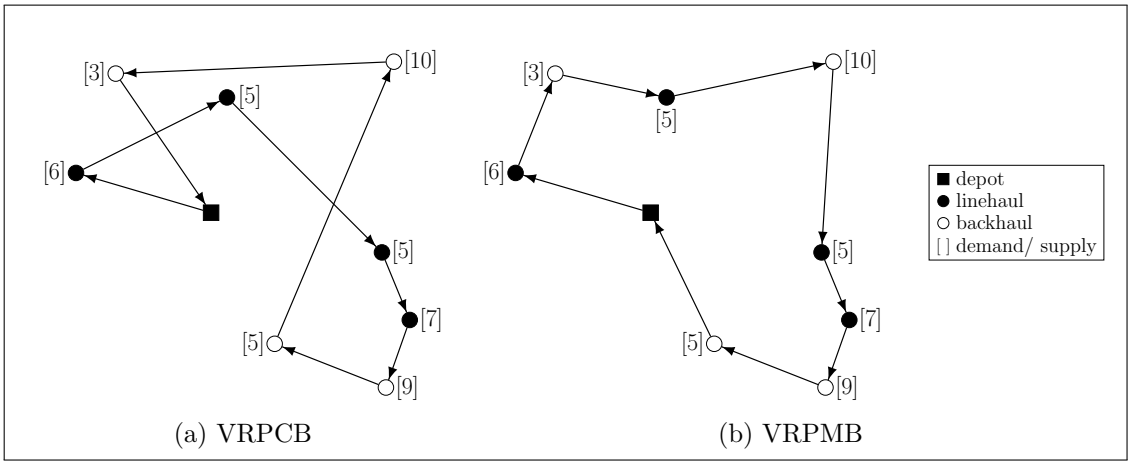


Figure 2.17: Instance with possible solutions for different VRPs with backhauls

Figure 2.18 shows the development of the weight of the items, that are inside the vehicle during the execution of the routes depicted in Figure 2.17. In the case of the VRPCB (Figure 2.18a), the cargo weight within the vehicle is constantly decreasing until all linehaul customers are served and then constantly increasing until all backhaul supplies are picked up. Hence, it is sufficient to check for the

adherence to the weight constraint at the beginning and at the end of the route. Similarly, if the non-overlapping constraint is satisfied at the beginning and the end of the route, it is also satisfied along the route due to the fact that any repositioning of the items is forbidden. In contrast, the cargo weight is fluctuating in the VRPMB route (Figure 2.18b) since goods can be loaded or unloaded at every stop. This aspect must be taken into account for the weight constraint (R3), which states that all *simultaneously transported items* must not exceed the permitted weight limit. Analogously, the simultaneous transport of linehaul and backhaul items needs to be regarded for the non-overlapping constraint (P2).

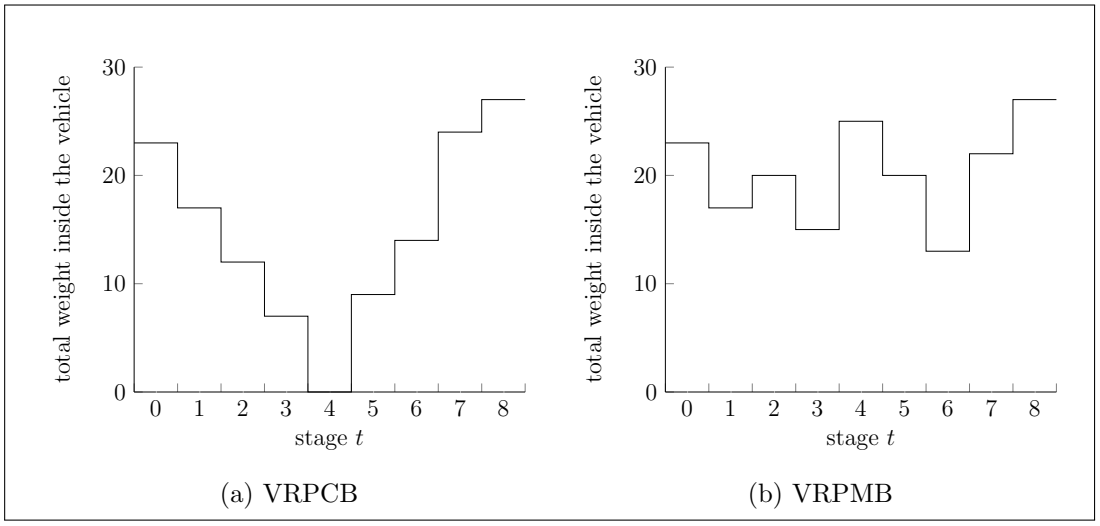


Figure 2.18: Development of the total weight in the vehicle for different vehicle routing problems with backhauls (example)

The LIFO constraint has to be considered among linehaul items only (P7a) and among backhauls items only (P7b) as before, but also regarding the positioning of linehaul items in relation to backhaul items. In this case, those items that are in the vehicle at the same time have to be taken into account. The adapted LIFO constraint can be formulated as follows:

(P7c) If a backhaul customer j ($j \in N^B$) is visited before a linehaul customer i ($i \in N^L$) in the same route, no item I_{jq} ($q \in J_j$) of customer j may be placed in front of, above, behind, or under any item I_{ik} ($k \in J_i$) of customer i.

The different relative positions are illustrated in Figure 2.19. If a backhaul item is collected before a linehaul item is delivered, it must not block the unloading of the linehaul item. Therefore, it must not be placed in front of (Figure 2.19a) or above

(Figure 2.19b) it. In addition, a backhaul item must not be placed behind (Figure 2.19c) or under (Figure 2.19d) a linehaul item because, otherwise, the linehaul item would need to be moved when the backhaul item is loaded.

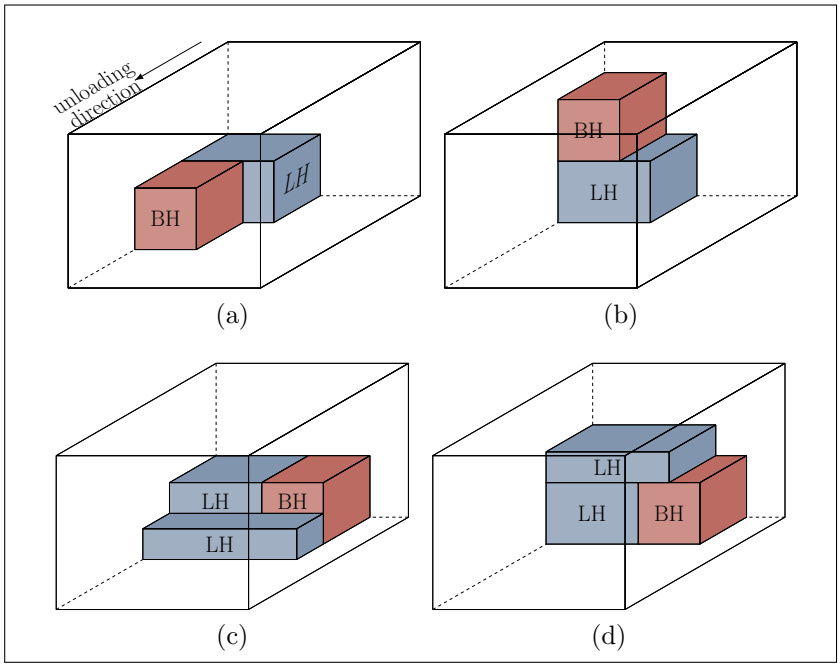


Figure 2.19: Infeasible placements of simultaneously transported linehaul (LH) and backhaul (BH) items

The remaining packing constraints (P1)-(P6), (P8)-(P10) can be applied unchanged. The LIFO constraint for the 3L-VRPMB(TW) facilitates the formulation of the problem to the effect that these constraints can be regarded for linehaul and backhaul items separately. For example, the fragility constraint does not have to be considered for a mixture of linehaul and backhaul items because they cannot be placed above one another due to (P7c).

The solution constraints (S1)-(S4) and the routing constraints (R1)-(R4) can be adopted from the general problem formulation.

Rear loaded vehicles (as presented above) are not used for the 3L-VRPMB(TW) as they make it difficult to arrange linehaul and backhaul items in a way that they do not block each other. Thus, in order to realize the simultaneous transport of linehaul and backhaul items, two alternative loading approaches are applied. The first approach consists in separating the loading space horizontally. For the sake of simplicity, it is assumed that the loading space is divided into two equal sections. Vehicles that are equipped with the possibility of separating the loading space horizontally are called double-decker vehicles. The separate sections can be used for the different item types (linehaul and backhaul) and the LIFO constraint

(P7c) does not need to be considered with respect to a mixture of the types. The approach is illustrated in Figure 2.20. The drawback of the loading space partition (LSP) approach is that the full volume of the loading space cannot be utilized throughout the whole route. The vehicles always leave and return to the depot with an (at least) half empty loading space.

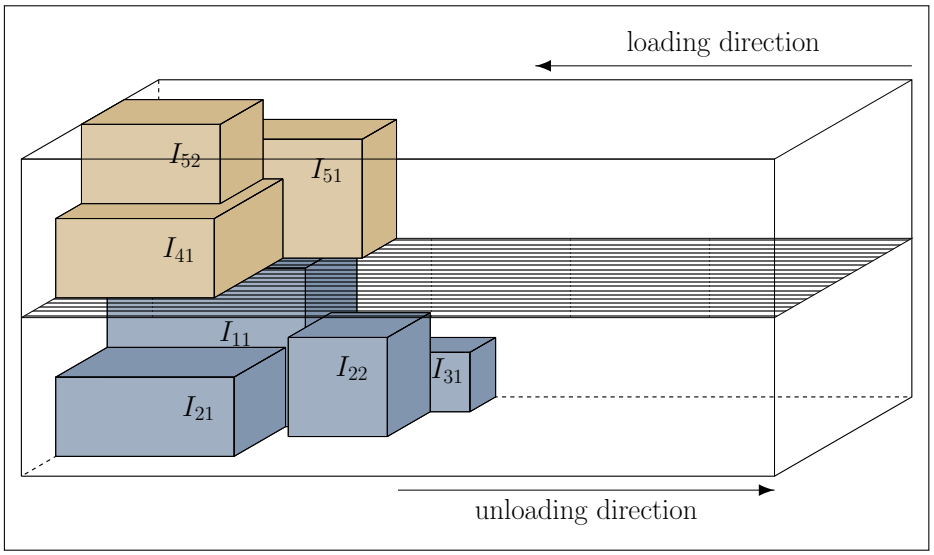


Figure 2.20: Illustration of the loading space partition approach

The side loading approach introduced for the 3L-VRPTW and 3L-VRPCB(TW) is also applied to the 3L-VRPMB(TW) allowing the full volume utilization. In contrast to the rear loading approach, the longer unloading side facilitates placing linehaul and backhaul items next to each other in a way that loading and unloading operations are not obstructed. As linehaul and backhaul items are placed in the same loading space, LIFO constraint (P7c) must be regarded.

A compact problem formulation could read as follows: Three-dimensional items need to be delivered from a central depot to a set of linehaul customers and picked up from a set of backhaul customers and brought to the depot using a given fleet of v_{max} homogeneous vehicles. Linehaul and backhaul customers can be visited in any sequence. A solution contains v_{used} tuples (R_v, PP_v). A tuple (R_v, PP_v) contains a route R_v for each used vehicle v ($v = 1, \ldots, v_{used}$) and the corresponding packing plan PP_v.

In order to be feasible, a solution must fulfil (S1)-(S4) and each route must adhere to (R1)-(R3) and, optionally, (R4) (time windows). The packing plans must satisfy (P1)-(P4), the LIFO constraint and the remaining relevant constraints of the respective constraint sets (cf. Table 2.1). The LIFO constraint (P7a) is applied to linehaul

items and (P7b) is applied to backhaul items. If linehaul and backhaul items are placed in the same loading space (i.e. not in separate sections), LIFO constraint (P7c) needs to be considered preventing, for example, backhaul items from blocking the unloading of linehaul items that are yet to be delivered. A feasible solution is to be determined that minimizes the TTD.
The following variants are regarded: The problem is considered with and without time windows, with the constraint sets C1-C5 and the loading approaches LSP and side loading.

2.1.2.4 The 3L-VRP with simultaneous delivery and pickup and time windows

The 3L-VRP with simultaneous delivery and pickup (and time windows; 3L-VRP-SDP(TW)) represents a generalization of the VRP with simultaneous delivery and pickup (VRPSDP), where each customer has both linehaul demands and backhaul supplies, and must be visited exactly once. That is, the delivery and pickup of items happens simultaneously. In contrast to the 3L-VRPCB(TW) and 3L-VRPMB(TW), not the set of customers is divided into two subsets but the respective item set of each customer. Let m_i^L be the number of linehaul items demanded by customer $i \in N_c$ and m_i^B the number of backhaul items supplied by customer i, i.e. $m_i = m_i^L + m_i^B$. The resulting subsets of J_i are the set of linehaul items $J_i^L = \{1, \ldots, m_i^L\}$ and the set of backhaul items $J_i^B = \{m_i^L + 1, \ldots, m_i\}$.
Similarly to the 3L-VRPMB(TW), linehaul and backhaul items can be in the vehicle at the same time. Therefore, the same considerations as above also apply here, i.e. all constraints have to be satisfied along each route and the majority of the packing constraints can be formulated for linehaul and backhaul items separately. The general LIFO constraint (P7) is adapted to the 3L-VRPSDP(TW):

(P7d) If customer j ($j \in N_c$) is visited after customer i ($i \in N_c$) in the same route,

(1) no linehaul item I_{jq} ($q \in J_j^L$) of customer j may be placed in front of or above any linehaul item I_{ik} ($k \in J_i^L$) of customer i,

(2) no backhaul item I_{jq} ($q \in J_j^B$) of customer j may be placed behind or under any backhaul item I_{ik} ($k \in J_i^B$) of customer i.

(3) no backhaul item I_{ik} ($k \in J_i^B$) of customer i may be placed in front of, above, behind or under any linehaul item I_{jq} ($q \in J_j^L$) of customer j.

The routing constraints (R1)-(R4) and solutions constraints (S1)-(S4) can be adopted.

As linehaul and backhaul items are transported simultaneously, the loading approaches applied to 3L-VRPMB(TW) – side loading and loading space partition – are applied to the 3L-VRPSDP(TW), too.

A compact problem formulation could read as follows: Three-dimensional items need to be delivered from a central depot to a set of customers and, at the same time, items have to be picked up from the customers using a given fleet of v_{max} homogeneous vehicles. A solution contains v_{used} tuples (R_v, PP_v). A tuple (R_v, PP_v) contains a route R_v for each used vehicle v ($v = 1, \ldots, v_{used}$) and the corresponding packing plan PP_v.

In order to be feasible, a solution must fulfil (S1)-(S4) and each route must adhere to (R1)-(R3) and, optionally, (R4) (time windows). The packing plans must satisfy (P1)-(P4), the LIFO constraint and the remaining relevant constraints of the respective constraint sets (cf. Table 2.1). The LIFO constraint (P7d) represents the general LIFO constraint (P7). If the LSP approach is applied, constraint (3) of (P7d) is automatically observed. A feasible solution is to be determined that minimizes the TTD.

The following variants are regarded: The problem is considered with and without time windows, with the constraint sets C1-C5 and the loading approaches LSP and side loading.

2.1.2.5 Overview of the considered problem variants

In the following, a problem variant of the 3L-VRPBTW is defined by the routing problem with respect to the backhauls and time windows. That is, the seven problem variants 3L-VRPTW, 3L-VRPCB, 3L-VRPCBTW, 3L-VRPMB, 3L-VRPMBTW, 3L-VRPSDP and 3L-VRPSDPTW are considered.

Furthermore, let an *extended* problem variant be additionally defined by its loading approach and constraint set. All considered extended variants are listed in Table 2.2. In total, 70 different combinations are taken into account. For example, the 3L-

VRPTW is exclusively regarded *with* time windows. Two loading approaches (rear and side loading) and five constraint sets are considered, adding up to ten different variants ($1 \cdot 2 \cdot 5$).

Table 2.2: Overview of the extended problem variants

problem variant	**time windows**		**loading approaches**			**constraint sets**				
	without	with	RL	LSP	SL	C1	C2	C3	C4	C5
3L-VRPTW		✓	✓		✓	✓	✓	✓	✓	✓
3L-VRPCB(TW)	✓	✓	✓		✓	✓	✓	✓	✓	✓
3L-VRPMB(TW)	✓	✓		✓	✓	✓	✓	✓	✓	✓
3L-VRPSDP(TW)	✓	✓		✓	✓	✓	✓	✓	✓	✓

LSP: loading space partition, RL: rear loading, SL: side loading, C1-C5: see Table 2.1

In the following, an extended problem variant is denoted by a triple containing the problem variant, the loading approach and the constraint set, e.g. (3L-VRPTW, RL, C1). In this context, the loading approaches are abbreviated as RL (rear loading) and SL (side loading), and (as usual in the text) LSP (loading space partition). If two extended problem variants share a given feature, they can also be merged in one triple. For example, (3L-VRPMB, LSP/SL, C1) refers to the extended problem variant of the 3L-VRPMB with C1 and with either the LSP approach or side loading.

2.2 Mathematical model

In the following, mathematical models for the problems described above are presented. First, the model for the 3L-VRPTW is presented in detail (Chapter 2.2.1). It serves as the basis for the models with backhauls, which are introduced in Chapters 2.2.2 to 2.2.4.

2.2.1 The 3L-VRP with time windows

The following models are based on the model introduced by Junqueira et al. (2013) for the 3L-CVRP. In this model, a time-dependent formulation is deployed, which is proposed by Fox et al. (1980) for the travelling salesman problem (TSP). The advantage of a time-dependent formulation is to have decision variables available that provide information about the sequence of customers within a route. This is, for example, helpful for formulating the LIFO constraint, which requires information about the loading and unloading sequence of the items. Moreover, the modelling of

the packing subproblem is based on formulations for the container loading problem (CLP) previously proposed by Junqueira et al. (2012a,b).

The model proposed by Junqueira et al. (2013) is extended by several constraints, which have not been considered in it. These constraints concern the time windows, the robust stability (as an extension of the proposed formulation for the vertical stability constraint), and reachability. Moreover, the fragility, load bearing strength and LIFO constraints are newly formulated since they have been approached differently in the work of Junqueira et al. (2013). Finally, also the possibility of rotating an item horizontally is implemented in the model.

Sets and constants

In the following, the constants and sets introduced in Chapter 2.1 are utilized. They are summarized in Table 2.3.

Table 2.3: Sets and constants for the mathematical model

group	**symbol**	**definition**
sets	E	edge set ($E = \{(i,j) \mid i,j \in N\}$)
	J_i	set of items demanded by customer $i \in N_c$ ($J_i = \{1,\ldots,m_i\}$)
	N	node set ($N = \{0,1,\ldots,n\}$)
	N_c	set of customers ($N_c = \{1,2,\ldots,n\}$)
	V	vehicle set ($V = \{1,\ldots,v_{max}\}$)
constants (instance)	n	number of customers
	v_{max}	number of available vehicles
	VSP	vertical stability parameter
	λ	maximum reach
constants (customers)	c_{ij}	cost of edge $(i,j) \in E$
	DD_i	due date of location $i \in N$
	m_i	number of items demanded by customer $i \in N_c$
	RT_i	ready time of location $i \in N$
	ST_i	service time of location $i \in N$
constants (items)	d_{ik}	weight of item I_{ik} ($i \in N_c, k \in J_i$)
	f_{ik}	fragility flag of item I_{ik} ($i \in N_c, k \in J_i$)
	h_{ik}	height of item I_{ik} ($i \in N_c, k \in J_i$)
	l_{ik}	length of item I_{ik} ($i \in N_c, k \in J_i$)
	p_{ik}	load bearing strength of item I_{ik} ($i \in N_c, k \in J_i$)
	w_{ik}	width of item I_{ik} ($i \in N_c, k \in J_i$)
constants (vehicles)	D	vehicle weight capacity
	H	vehicle loading space height
	L	vehicle loading space length
	W	vehicle loading space width

Decision variables

The model contains three kinds of decision variables: routing variables, packing variables and orientation variables. The binary routing decision variables x_{ij}^{tv} $(i, j, t \in N, v \in V)$ are defined as

$$x_{ij}^{tv} = \begin{cases} 1, & \text{if vehicle } v \text{ goes directly from node } i \text{ to node } j \text{ in stage } t, \\ 0, & \text{otherwise.} \end{cases} \tag{2.8}$$

A stage refers to the section of the route in which node i is left and node j is approached. In stage $t = 0$, the depot is left and the first customer in a route is visited. The variables are illustrated for one route in Figure 2.21.

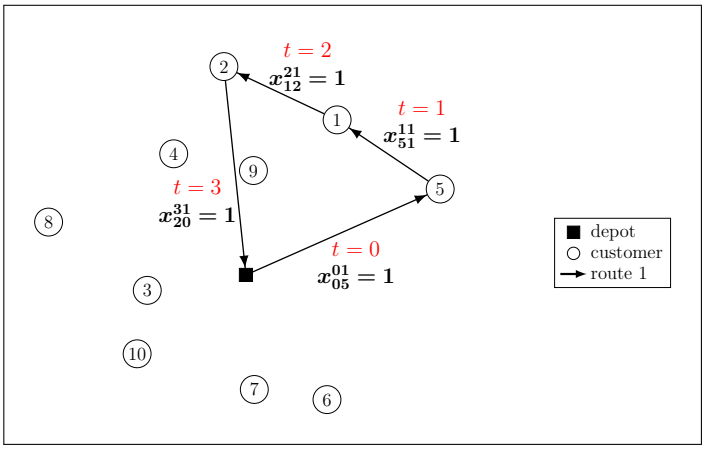


Figure 2.21: Partial VRP solution with variables

Secondly, binary placement decision variables represent the coordinates where an item is placed within the loading space with its back-left-bottom corner point. As mentioned above, the loading space can be embedded in a three-dimensional Cartesian coordinate system (see Figure 2.1). Thus, the back-left-bottom corner point of an item is defined as the point that is closest to the origin of the coordinate system. Let A', B' and Γ' be the sets of possible positions for the corner points along the α-, β- and γ-axes without considering rotations:

$$A' = \left\{0, 1, 2, \ldots, L - \min_{i \in N_c, k \in J_i} l_{ik}\right\}, \tag{2.9}$$

$$B' = \left\{0, 1, 2, \ldots, W - \min_{i \in N_c, k \in J_i} w_{ik}\right\} \text{ and} \tag{2.10}$$

$$\Gamma' = \left\{0, 1, 2, \ldots, H - \min_{i \in N_c, k \in J_i} h_{ik}\right\}. \tag{2.11}$$

These sets can be restricted in the shown way since all the points that are, e.g., above the height $H - \min_{i \in N_c, k \in J_i} h_{ik}$ cannot be the corner points of any item. Being placed there, the items would overlap with the container walls. Moreover, the potential placement coordinates are exclusively assumed to be integer since all item dimensions are also assumed to be integer.

Furthermore, let

$$A'_{ik} = \{\alpha \in A | 0 \leq \alpha \leq L - l_{ik}\}, \tag{2.12}$$

$$B'_{ik} = \{\beta \in B | 0 \leq \beta \leq W - w_{ik}\} \text{ and} \tag{2.13}$$

$$\Gamma'_{ik} = \{\gamma \in \Gamma | 0 \leq \gamma \leq H - h_{ik}\}, \tag{2.14}$$

be subsets of A', B' and Γ' referring to the individual items I_{ik} $(i \in N_c, k \in J_i)$. Restricting the sets of coordinates for the placement in the shown way would ensure that each item is placed completely inside the loading space if rotation was not allowed.

Considering the possible rotation of items on the horizontal plane, the possible coordinate sets for the corner points are the following:

$$A = \left\{0, 1, 2, \ldots, L - \min\left(\min_{i \in N_c, k \in J_i} l_{ik}, \min_{i \in N_c, k \in J_i} w_{ik}\right)\right\}, \tag{2.15}$$

$$B = \left\{0, 1, 2, \ldots, W - \min\left(\min_{i \in N_c, k \in J_i} l_{ik}, \min_{i \in N_c, k \in J_i} w_{ik}\right)\right\} \text{and} \tag{2.16}$$

$$\Gamma = \left\{0, 1, 2, \ldots, H - \min_{i \in N_c, k \in J_i} h_{ik}\right\}; \tag{2.17}$$

and the subsets for the individual items I_{ik} $(i \in N_c, k \in J_i)$ can be formulated as:

$$A_{ik} = \{\alpha \in A | 0 \leq \alpha \leq L - \min(l_{ik}, w_{ik})\}, \tag{2.18}$$

$$B_{ik} = \{\beta \in B | 0 \leq \beta \leq W - \min(l_{ik}, w_{ik})\} \text{and} \tag{2.19}$$

$$\Gamma_{ik} = \{\gamma \in \Gamma | 0 \leq \gamma \leq H - h_{ik}\}. \tag{2.20}$$

Hence, if rotation is permitted, further restrictions need to be formulated in order to ensure the complete placement within the vehicle. They are presented below.

The placement decision variables

$$\pi^{iktv}_{\alpha\beta\gamma}, \ \forall\, i \in N_c, k \in J_i, t \in N \setminus \{n\}, v \in V, \alpha \in A_{ik}, \beta \in B_{ik}, \gamma \in \Gamma_{ik},$$

are defined as

$$\pi_{\alpha\beta\gamma}^{iktv} = \begin{cases} 1, & \text{if item } I_{ik} \text{ of customer } i \text{ who is visited in stage } t \\ & \text{by vehicle } v \text{ is placed with its back-left-bottom} \\ & \text{corner at position } (\alpha, \beta, \gamma), \\ 0, & \text{otherwise.} \end{cases} \tag{2.21}$$

Note that this variable is neither defined for $i = 0$ since the depot does not have any demands, nor for $t = n$. If a route would include all n customers, the n^{th} stage would be the trip back to the depot. Thus, for $t = n$ the variable $\pi_{\alpha\beta\gamma}^{iktv}$ would refer to $i = 0$.

With the help of these binary variables, the back-left-bottom corner point $(\alpha_{ik}, \beta_{ik}, \gamma_{ik})$ of item I_{ik} within the loading space can be determined as:

$$\alpha_{ik} = \sum_{t \in N \setminus \{n\}} \sum_{v \in V} \sum_{\alpha \in A_{ik}} \sum_{\beta \in B_{ik}} \sum_{\gamma \in \Gamma_{ik}} \alpha \cdot \pi_{\alpha\beta\gamma}^{iktv}, \tag{2.22}$$

$$\beta_{ik} = \sum_{t \in N \setminus \{n\}} \sum_{v \in V} \sum_{\alpha \in A_{ik}} \sum_{\beta \in B_{ik}} \sum_{\gamma \in \Gamma_{ik}} \beta \cdot \pi_{\alpha\beta\gamma}^{iktv}, \tag{2.23}$$

$$\gamma_{ik} = \sum_{t \in N \setminus \{n\}} \sum_{v \in V} \sum_{\alpha \in A_{ik}} \sum_{\beta \in B_{ik}} \sum_{\gamma \in \Gamma_{ik}} \gamma \cdot \pi_{\alpha\beta\gamma}^{iktv}. \tag{2.24}$$

The third kind of decision variables describes the chosen spatial orientation of an item. For each item I_{ik} $(i \in N_c, k \in J_i)$, the orientation is represented by the binary variable o_{ik}, where

$$o_{ik} = \begin{cases} 1, & \text{if the length } l_{ik} \text{ of item } I_{ik} \text{ is parallel to the } \alpha\text{-axis,} \\ 0, & \text{otherwise.} \end{cases} \tag{2.25}$$

Moreover, in order to achieve more clarity in the model, the variables $\widehat{l}_{ik}$ and $\widehat{w}_{ik}$ represent the dimensions of an item I_{ik} $(i \in N_c, k \in J_i)$ with respect to the chosen orientation. $\widehat{l}_{ik}$ refers to the dimension of item I_{ik} that is parallel to the α-axis, i.e.

$$\widehat{l}_{ik} = o_{ik} \cdot l_{ik} + (1 - o_{ik}) \cdot w_{ik}. \tag{2.26}$$

Analogously, $\widehat{w}_{ik}$ represents the dimension of I_{ik} that is parallel to the β-axis, i.e.

$$\widehat{w}_{ik} = (1 - o_{ik}) \cdot l_{ik} + o_{ik} \cdot w_{ik}. \tag{2.27}$$

Model

With the defined decision variables and the data introduced above, the mathematical model for the 3L-VRPTW is formulated. The objective function consists of the minimization of the total routing cost, i.e. the total travel distance (TTD):

$$\min z = \sum_{i \in N} \sum_{j \in N} \sum_{t \in N} \sum_{v \in V} c_{ij} \cdot x_{ij}^{tv} \tag{2.28}$$

Routing constraints

The following routing constraints have to be satisfied:

$$\sum_{j \in N} \sum_{t \in N} \sum_{v \in V} x_{ij}^{tv} = 1 \qquad \forall\ i \in N_c, \tag{2.29}$$

$$\sum_{j \in N_c} x_{0j}^{0v} \leq 1 \qquad \forall\ v \in V, \tag{2.30}$$

$$x_{0j}^{tv} = 0 \qquad \forall\ j \in N_c, t \in N \setminus \{0\}, v \in V, \tag{2.31}$$

$$x_{ii}^{tv} = 0 \qquad \forall\ i \in N, t \in N, v \in V, \tag{2.32}$$

$$\sum_{j \in N} \sum_{t \in N_c} \sum_{v \in V} t \cdot x_{ij}^{tv} - \sum_{j \in N} \sum_{t \in N} \sum_{v \in V} t \cdot x_{ji}^{tv} = 1 \qquad \forall\ i \in N_c, \tag{2.33}$$

$$\sum_{j \in N} x_{ij}^{(t+1)v} - \sum_{j \in N} x_{ji}^{tv} = 0 \qquad \forall\ i \in N_c, t \in N \setminus \{n\}, v \in V. \tag{2.34}$$

Constraint (2.29)[8] guarantees that each customer is left exactly once. In connection with later constraints, it also ensures that each customer is only approached once. The depot can be left more than once, though, since it is left by up to v_{max} vehicles. Therefore, it is excluded in this constraint.

Constraint (2.30) ensures that each vehicle leaves the depot at most once in stage 0. Moreover, it is assumed that the vehicles leave the depot in stage 0 and not later (2.31). This constraint is also important for the validity of the constraints that follow.

[8] Technically, (2.29) represents multiple constraints of the optimization model. Nonetheless, it is referred to in the text as one constraint (in the singular). This format will also be maintained in the remainder of this thesis.

Constraint (2.32) guarantees that no vehicle approaches the same node directly after having left it, i.e. travelling the directed edge $(i, i)\ \forall\ i \in N$. In doing so, (2.32) also prevents the creation of routes without any customer locations as $x_{00}^{tv} = 1$ ($t \in N, v \in V$) is forbidden. Using this formulation, it can be assumed that $c_{ii} \in \mathbb{R}_0^+$ ($i \in N$).
Constraint (2.33) ensures the connectivity of routes. It guarantees that each customer $i \in N_c$ is left exactly one stage after being visited. In doing so, the constraint also ensures that the depot is included in each route and that all routes are circular. In other words, it prevents the building of subroutes not containing the depot. Note that the vehicle that visits and leaves a customer is neglected here as a customer being visited and left by the same vehicle is ensured by (2.34) (see below).[9]
Let, for example, a route be $R_v = (1, 2, 3, 4, 1)$, thus, without the depot. Vehicle v travels from customer 1 to customer 2 in stage 0.[10] Then – resulting from (2.34) – customer 3 must be visited in stage $t = 1$, customer 4 in stage $t = 2$, and customer 1 again in stage $t = 3$, i.e. $x_{12}^{0v} = x_{23}^{1v} = x_{34}^{2v} = x_{41}^{3v} = 1$. However, constraint (2.33) would lead to a contradiction for $i = 1$ (Fox et al., 1980):

$$\sum_{j \in N} \sum_{t \in N_c} \sum_{v \in V} t \cdot x_{1j}^{tv} - \sum_{j \in N} \sum_{t \in N} \sum_{v \in V} t \cdot x_{j1}^{tv} = 0 \cdot x_{12}^{0v} - 3 \cdot x_{41}^{3v} = -3 \neq 1.$$

Since node 0 is the only node that does not need to fulfil constraint (2.33), all routes must include the depot.
In another example, let $R_1 = (1, 2, 3, 4, 0)$ and $R_2 = (0, 1, 2, 3, 4)$. The routes include the depot but are not circular. In R_1, customer 1 is left in $t = 0$. As this customer is not visited by any vehicle beforehand, this route violates (2.33). The term on the left-hand side would equal 0. In R_2, the depot is left in $t = 0$ (due to (2.31)). Thus, customer 4 is visited in $t = 3$. As it is never left, the left-hand side of (2.33) would equal 3 for $i = 4$ violating the constraint.
Constraint (2.34) represents the so-called flow conservation constraint, i.e. if vehicle v travels to customer i in stage t, then vehicle v must leave customer i in stage $t + 1$. Although constraints (2.33) and (2.34) might seem similar at first sight, they are both vital to the model, which should be illustrated in the following example: Figure 2.22 shows the routes of three different vehicles in the optimal solution for a

[9] Reformulating (2.33) so that it is defined not only for a specific user $i \in N_c$ but also for a specific vehicle $v \in V$ would not work. The constraint would be violated for every vehicle that does *not* contain i in its route as the left-hand side would equal 0.
[10] Being left later would violate (2.34).

CVRP with ten customers.[11]

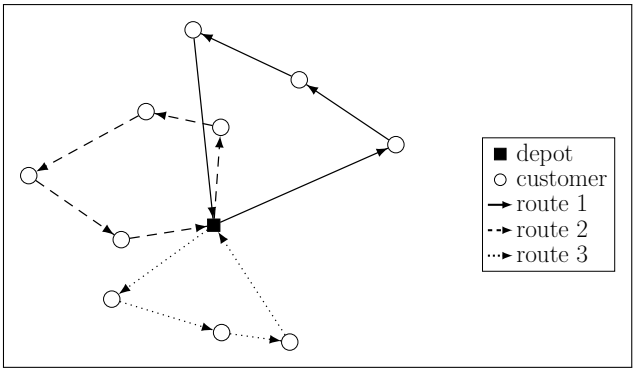
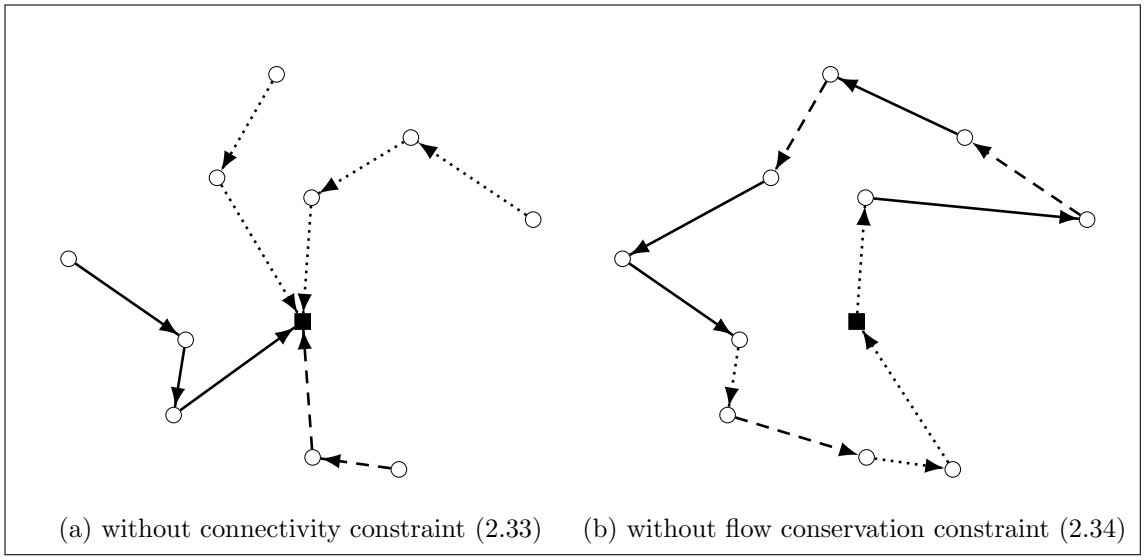


Figure 2.22: Optimal solution for a CVRP instance

Figure 2.23 shows optimal solutions for the same instance. However, here constraint (2.33) (in Figure 2.23a) and constraint (2.34) (in Figure 2.23b), respectively, are excluded from the model.



(a) without connectivity constraint (2.33) (b) without flow conservation constraint (2.34)

Figure 2.23: CVRP solutions excluding constraints (2.33) and (2.34), respectively

As can be seen from Figure 2.23a, without the connectivity constraint (2.33) unconnected routes are formed that do not start and end at the depot. In that case, (2.34) is still fulfilled, as each customer visited by a certain vehicle in a stage $t \geq 0$ is left by that vehicle in a stage $t+1$. The locations at the beginning of a route, i.e. those that are left in stage $t = 0$, are excluded. Since they are left in $t = 0$, they would need to be approached in stage $t = -1$ for which the constraint is not defined. Thus, unconnected routes can occur because the flow conservation constraint (2.34)

[11] A small CVRP instance with ten customers was generated and solved using CPLEX. As it serves only the purpose of illustration, providing the detailed instance data is omitted here.

alone does not guarantee that each customer who is left in stage t is visited in stage $t-1$. This is done by (2.33).

Without the flow conservation constraint (Figure 2.23b), each customer is visited in a stage and left in the next stage; however, not by the same vehicle.

Furthermore, the capacity constraints of the one-dimensional CVRP can be formulated as weight (2.35) and volume constraints (2.36) for the 3L-VRPTW:

$$\sum_{i \in N_c} \sum_{k \in J_i} \sum_{j \in N} \sum_{t \in N_c} d_{ik} \cdot x_{ij}^{tv} \leq D \qquad \forall\, v \in V, \tag{2.35}$$

$$\sum_{i \in N_c} \sum_{k \in J_i} \sum_{j \in N} \sum_{t \in N_c} l_{ik} \cdot w_{ik} \cdot h_{ik} \cdot x_{ij}^{tv} \leq L \cdot W \cdot H \qquad \forall\, v \in V. \tag{2.36}$$

On the left-hand sides of the constraints, the weights or volumes of *all* items transported in a route v are summed. The development of the transported weights and volumes in the course of the route does not need to be considered as the loaded weights and volumes constantly decrease during the route. Thus, all items placed in the loading space at the beginning of the route must not violate the capacity constraints. As the remaining routing constraints ensure that each customer is left exactly once, each customer is caputured in (2.35) and (2.36).

Constraint (2.37) ensures that the routing variables are bound to the packing variables, i.e. if vehicle v visits customer i in stage t, vehicle v has to transport all items of customer i and unload them in stage t:

$$\sum_{k \in J_i} \sum_{\alpha \in A_{ik}} \sum_{\beta \in B_{ik}} \sum_{\gamma \in \Gamma_{ik}} \pi_{\alpha\beta\gamma}^{iktv} = m_i \cdot \sum_{j \in N} x_{ji}^{tv} \qquad \forall\, i \in N_c, t \in N \setminus \{n\}, v \in V. \tag{2.37}$$

Finally, constraint (2.38) ensures that every item is packed exactly once:

$$\sum_{t \in N \setminus \{n\}} \sum_{v \in V} \sum_{\alpha \in A_{ik}} \sum_{\beta \in B_{ik}} \sum_{\gamma \in \Gamma_{ik}} \pi_{\alpha\beta\gamma}^{iktv} = 1 \qquad \forall\, i \in N_c, k \in J_i. \tag{2.38}$$

As any reloading during the execution of a route is forbidden, each item is assigned exactly one placement. Thus, for each item I_{ik} ($i \in N_c, k \in J_i$), exactly one variable $\pi_{\alpha\beta\gamma}^{iktv}$ ($t \in N \setminus \{n\}, v \in V, \alpha \in A_{ik}, \beta \in B_{ik}, \gamma \in \Gamma_{ik}$) must assume the value 1.

The following restrictions ensure the adherence of the time windows. For this purpose, the formulations proposed by Moura and Oliveira (2009) are extended to the time-dependent formulation. Hereby, let $start_{iv}$ be the time at which vehicle v

begins serving customer i, and M_1 a sufficiently large number ("*Big-M*"):

$$c_{0i} - start_{iv} \leq M_1 \cdot (1 - x_{0i}^{0v}) \qquad \forall\, i \in N_c, v \in V, \tag{2.39}$$

$$start_{iv} + ST_i + c_{ij} - start_{jv} \leq M_1 \cdot (1 - x_{ij}^{tv}) \qquad \forall\, i \in N_c, j \in N, v \in V, t \in N_c, \tag{2.40}$$

$$start_{iv} \geq RT_i \qquad \forall\, i \in N, v \in V, \tag{2.41}$$

$$start_{iv} \leq DD_i \qquad \forall\, i \in N, v \in V. \tag{2.42}$$

It is assumed that every route starts at time zero. Constraint (2.39) refers to the arrival at the customer locations that are visited in stage $t = 0$, i.e. if customer i is visited in the 0^{th} stage (directly at the beginning of the route), then $x_{0i}^{0v} = 1$. Hence, the arrival time at customer i equals the travel time between customer i and the depot (c_{0i}). In addition, (2.39) ensures that the service at customer i does not start before the arrival at the location because $x_{0i}^{0v} = 1$ results in $c_{0i} \leq start_{iv}$. In connection with (2.41), it is guaranteed that the service does not start before the customer's ready time RT_i. Thus, the formulations allow the consideration of waiting times between arrival and start of service.

Similarly, constraint (2.40) guarantees that the services at locations that are visited later than stage $t = 0$ do not start before the arrivals at the locations. Let customer j be visited directly after customer i by vehicle v. In this case, the arrival time at customer j depends on the start of service at customer i ($start_{iv}$), the service time at i (ST_i) and the travel time between i and j (c_{ij}). It must, thus, be fulfilled that

$$start_{jv} \geq start_{iv} + ST_{iv} + c_{ij}. \tag{2.43}$$

As $start_{0v}$ refers to the arrival time at the depot at the *end* of each route, constraint (2.40) cannot be formulated for $t = 0$, which is why (2.39) needs to be formulated separately.

If customer i is *not* visited by vehicle v, the formulations allow $start_{iv}$ to accept any value due to the usage of the *Big-M*. In order to be large enough and to secure the *worst case*, the following magnitude should be chosen for M_1:

$$M_1 \geq \max_{i \in N} DD_i + \max_{i \in N} ST_i + \max_{(i,j) \in E} c_{ij} - \min_{i \in N} RT_i. \tag{2.44}$$

The adherence of the time windows at every location is ensured by (2.41) and (2.42). Set V contains v_{max} vehicles. Thus, the tour number restriction constraint (S4) is always satisfied due to the definitions of the routing constraints for all vehicles $v \in V$.

Packing constraints

In addition to a routing plan, a feasible packing plan must exist for each route satisfying the constraints described in Chapter 2.1.

Although other loading approaches, like side loading or separation of the loading space, are also considered in this thesis, the following constraints are only formulated for the standard rear loading approach. They can be adapted easily to the other approaches, though.

Geometrical constraints (P1)-(P3)

One aspect of the geometrical constraints is to ensure that all items are placed completely within the loading space and can be stated as:

$$\sum_{\left\{\alpha \in A_{ik} | \alpha > L - \widehat{l}_{ik}\right\}} \sum_{\beta \in B_{ik}} \pi_{\alpha\beta\gamma}^{iktv} + \sum_{\alpha \in A_{ik}} \sum_{\left\{\beta \in B_{ik} | \beta > W - \widehat{w}_{ik}\right\}} \pi_{\alpha\beta\gamma}^{iktv} = 0 \qquad (2.45)$$

$$\forall\, i \in N_c, k \in J_i, t \in N \setminus \{n\}, v \in V, \gamma \in \Gamma_{ik}.$$

The first summand on the left-hand side sums the values of the placement decision variables referring to item I_{ik} and $\alpha > L - \widehat{l}_{ik}$. The second summand sums the values of the placement decision variables referring to item I_{ik} and $\beta > W - \widehat{w}_{ik}$. That is, the corner point coordinate α_{ik} of item I_{ik} ($i \in N_c, k \in J_i$) cannot take any value greater than $L - \widehat{l}_{ik}$ anywhere along the width and height axes, neither can the coordinate β_{ik} take any value greater than $W - \widehat{w}_{ik}$ anywhere along the length and height axes. As mentioned before, this constraint is not necessary if items cannot be rotated. In this case, the placement inside the vehicle is guaranteed by the restricted coordinate sets A'_{ik} and B'_{ik}. Since the height dimension of an item is fixed, the restricted definition of Γ_{ik} (which is equal to Γ'_{ik}) is sufficient to ensure that an item is not placed above a feasible height level.

The second kind of geometrical constraints ensures the non-overlapping of the items. More precisely, it is guaranteed that every point $(\alpha', \beta', \gamma')$ in the loading space is

occupied by not more than one item. Item I_{ik} having the back-left-bottom corner point (α, β, γ) occupies the point $(\alpha', \beta', \gamma')$ if the following is true:

$$\alpha \leq \alpha' \leq \alpha + \widehat{l}_{ik} - 1 < L, \tag{2.46}$$

$$\beta \leq \beta' \leq \beta + \widehat{w}_{ik} - 1 < W \text{ and} \tag{2.47}$$

$$\gamma \leq \gamma' \leq \gamma + h_{ik} - 1 < H. \tag{2.48}$$

That is, if item I_{ik} has its back-left-bottom corner point at (α, β, γ), the points it occupies have α-coordinates from α to $\alpha + \widehat{l}_{ik} - 1$ (analogously for the β- and γ-coordinates). Thus, the points at the front, right and top face of an item do not count as occupied by that item.

Figure 2.24 shows the β-γ-projection of three arranged items and the points that are occupied by the respective items. As can be seen, the points at the right side of item I_{11} (at $\beta = 5$) are not occupied by I_{11}, but by the adjacent item I_{31}. The same accounts for its top face ($\gamma = 5$).

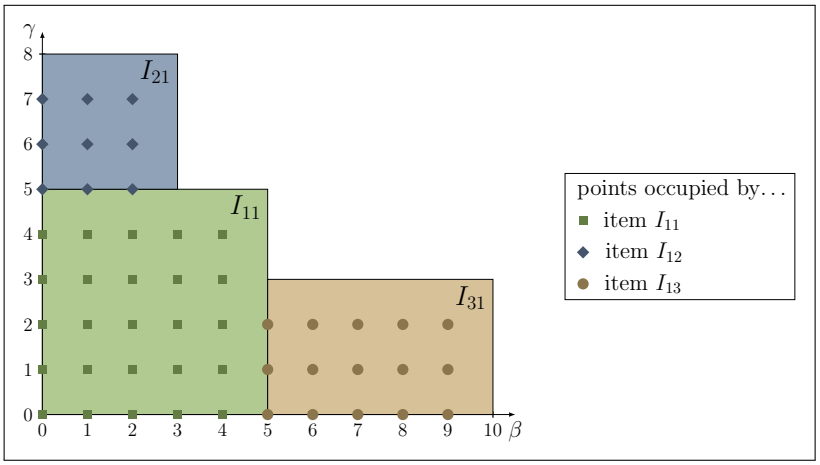


Figure 2.24: Illustration of occupied points

The other way around, it can be stated that a point $(\alpha', \beta', \gamma')$ is occupied by item I_{ik} with its back-left-bottom corner point at $(\alpha_{ik}, \beta_{ik}, \gamma_{ik})$, if

$$\alpha' - \widehat{l}_{ik} + 1 \leq \alpha_{ik} \leq \alpha', \tag{2.49}$$

$$\beta' - \widehat{w}_{ik} + 1 \leq \beta_{ik} \leq \beta' \text{ and} \tag{2.50}$$

$$\gamma' - h_{ik} + 1 \leq \gamma_{ik} \leq \gamma'. \tag{2.51}$$

Figure 2.25 illustrates how to determine whether a point $(\alpha', \beta', \gamma')$ is occupied by an item. The figure shows a two-dimensional (2D) projection of two items. It is

assumed that both items have the same height h and stand on the container floor. The point $(\alpha', \beta', \gamma')$ is at $(4, 4, \gamma')$ with $0 \leq \gamma' < h$. The "search regions" include the points where the back-left-bottom corner points of the respective item would need to be so that the item occupies $(\alpha', \beta', \gamma')$. Since the item dimensions and placement coordinates are assumed to be integer, the search regions consist of finite sets of points. As can be seen, the corner point of item I_{21} is within the search region for I_{21}, i.e. the point $(\alpha', \beta', \gamma')$ is occupied by item I_{21}.

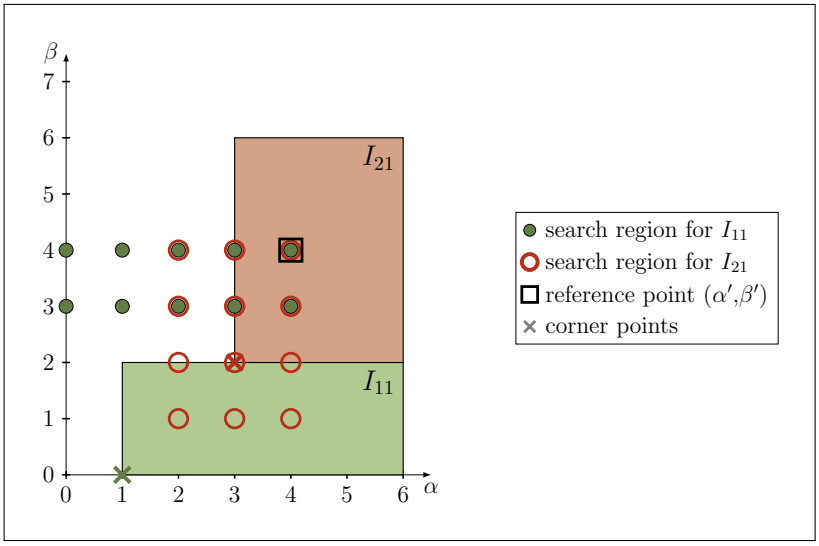


Figure 2.25: Illustration of occupied points and search regions

As described above, the front, right and top faces of an item are not considered to be occupied by the respective item. Therefore, these faces of the loading space do not need to be considered for the non-overlapping constraint either.

Let $A_0 = \{0, 1, ..., L-1\}, B_0 = \{0, 1, ..., W-1\}$ and $\Gamma_0 = \{0, 1, ..., H-1\}$. The non-overlapping constraint can, thus, be stated as:

$$\begin{aligned} &\sum_{i \in N_c} \sum_{k \in J_i} \sum_{t \in N \setminus \{n\}} \\ &\sum_{\{\alpha \in A_{ik} | \alpha' - \widehat{l}_{ik} + 1 \leq \alpha \leq \alpha'\}} \\ &\sum_{\{\beta \in B_{ik} | \beta' - \widehat{w}_{ik} + 1 \leq \beta \leq \beta'\}} \\ &\sum_{\{\gamma \in \Gamma_{ik} | \gamma' - h_{ik} + 1 \leq \gamma \leq \gamma'\}} \pi^{iktv}_{\alpha\beta\gamma} \leq 1 \\ &\forall\, v \in V, \alpha' \in A_0, \beta' \in B_0, \gamma' \in \Gamma_0. \end{aligned} \tag{2.52}$$

That is, the sum of items occupying a given point $(\alpha', \beta', \gamma')$ in vehicle v must not

exceed 1. The relations (2.49)-(2.51) are applied for the summation indices on the left-hand side.

Alternatively, new variables can be introduced describing whether a given point is occupied by a certain item. Although the non-overlapping constraint can be formulated without these variables, they can (i) make the constraint clearer and (ii) are needed later for the formulation of other constraints. Thus, let

$$\varphi_{\alpha\beta\gamma}^{ikv} = \begin{cases} 1, & \text{if the point } (\alpha,\beta,\gamma) \text{ is occupied by item } I_{ik} \text{ in vehicle } v, \\ 0, & \text{otherwise.} \end{cases} \tag{2.53}$$

Applying (2.49)-(2.51), the occupation variables can be determined as:

$$\begin{aligned} \varphi_{\alpha\beta\gamma}^{ikv} = \sum_{t\in N\setminus\{n\}} \sum_{\left\{\alpha'\in A_{ik}\middle|\alpha-\widehat{l}_{ik}+1\le\alpha'\le\alpha\right\}} \sum_{\left\{\beta'\in B_{ik}\middle|\beta-\widehat{w}_{ik}+1\le\beta'\le\beta\right\}} \\ \sum_{\left\{\gamma'\in\Gamma_{ik}\middle|\gamma-h_{ik}+1\le\gamma'\le\gamma\right\}} \pi_{\alpha'\beta'\gamma'}^{iktv} \end{aligned} \tag{2.54}$$

$$\forall\ i\in N_c, k\in J_i, v\in V, \alpha\in A_0, \beta\in B_0, \gamma\in\Gamma_0.$$

Using these variables, (2.52) can be reformulated as:

$$\sum_{i\in N_c}\sum_{k\in J_i}\varphi_{\alpha\beta\gamma}^{ikv} \le 1 \qquad \forall\ v\in V, \alpha\in A_0, \beta\in B_0, \gamma\in\Gamma_0. \tag{2.55}$$

The orthogonal packing of each item I_{ik} is ensured by the definition of the decision variables $\pi_{\alpha\beta\gamma}^{iktv}$ $(i\in N_c, k\in J_i, t\in N\setminus\{n\}, v\in V)$.

Fixed vertical orientation (P4)

The fixed vertical orientation of each item I_{ik} is ensured by the definition of the decision variables o_{ik} $(i\in N_c, k\in J_i)$.

Vertical stability (P5)

In the following, the vertical stability constraint as it is applied by Gendreau et al. (2006) is presented. Below, it is extended to the robust stability constraint as it is defined by Ceschia et al. (2013).

The constraint demands a given threshold percentage VSP of an item's bottom face to be supported either by the container floor or by the top faces of other items. A

mathematical formulation must be able to identify the items that support another item. In addition, it must identify the area of the bottom side that is supported. Let I_{ik} and I_{jq} be two items in the same vehicle with their back-left-bottom corner points at $(\alpha_{ik}, \beta_{ik}, \gamma_{ik})$ and $(\alpha_{jq}, \beta_{jq}, \gamma_{jq})$, respectively (see (2.22)-(2.24)). In order that item I_{jq} *directly* supports (see p. 18) item I_{ik}, several conditions must hold. First of all, the top face of I_{jq} must be at the same height as the bottom face of I_{ik}:

$$\gamma_{ik} = \gamma_{jq} + h_{jq}. \tag{2.56}$$

Moreover, the α-β-projections of both items must overlap. Regarding the length dimension it must be fulfilled that:

$$\alpha_{ik} - \widehat{l}_{jq} + 1 \leq \alpha_{jq} \leq \alpha_{ik} + \widehat{l}_{ik} - 1, \tag{2.57}$$

which is visualized in Figure 2.26. The reference corner point (marked by the dots) of item I_{jq} would be outside of this range if I_{jq} was in the left-most (1) or in the right-most position (3), whereas the corner point would be within the range if I_{jq} was in the middle position (2). Thus, I_{jq} would support I_{ik} if I_{jq} was in position (2). Analogously, such relations can be applied to the width dimension. In conclusion, I_{jq} supports I_{ik} (at least partially) if (2.56), (2.57) and

$$\beta_{ik} - \widehat{w}_{jq} + 1 \leq \beta_{jq} \leq \beta_{ik} + \widehat{w}_{ik} - 1 \tag{2.58}$$

are fulfilled.

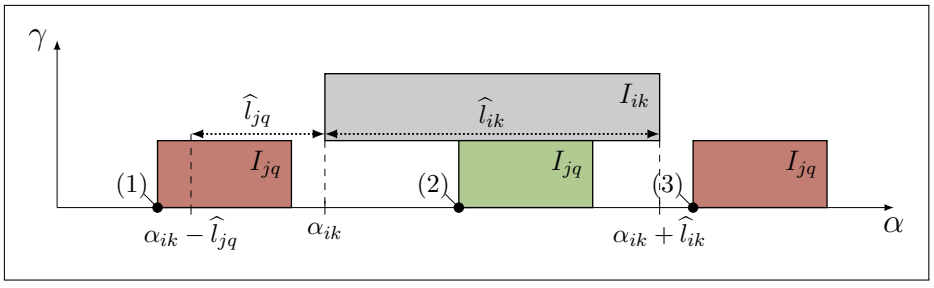


Figure 2.26: Relations of coordinates for determining whether an item is placed below another

Based on the remarks presented above, the vertical stability constraint can be stated as:

$$\sum_{j \in N_c} \sum_{q \in J_j} \sum_{\{u \in N | t \leq u < n\}} \sum_{\{\alpha \in A_{jq} | \alpha' - \widehat{l}_{jq} + 1 \leq \alpha \leq \alpha' + \widehat{l}_{ik} - 1\}} \sum_{\{\beta \in B_{jq} | \beta' - \widehat{w}_{jq} + 1 \leq \beta \leq \beta' + \widehat{w}_{ik} - 1\}} \tilde{l}_{ikjq} \cdot \widetilde{w}_{ikjq} \cdot \pi^{jquv}_{\alpha\beta(\gamma' - h_{jq})} \geq VSP \cdot l_{ik} \cdot w_{ik} \cdot \pi^{iktv}_{\alpha'\beta'\gamma'} \tag{2.59}$$

where

$$\tilde{l}_{ikjq} = \min(\alpha + \widehat{l}_{jq}, \alpha' + \widehat{l}_{ik}) - \max(\alpha, \alpha') \text{ and} \tag{2.60}$$

$$\widetilde{w}_{ikjq} = \min(\beta + \widehat{w}_{jq}, \beta' + \widehat{w}_{ik}) - \max(\beta, \beta') \tag{2.61}$$

$$\forall\, i \in N_c, k \in J_i, t \in N \setminus \{n\}, v \in V, \alpha' \in A_{ik}, \beta' \in B_{ik}, \gamma' \in \Gamma_{ik} \setminus \{0\}.$$

$\tilde{l}_{ikjq} \cdot \widetilde{w}_{ikjq}$ is the area, that an item I_{jq} provides to support item I_{ik}. Figure 2.27 illustrates a 2D projection on the α-β-plane and shows the support area provided by two different items. Item I_{11} is placed on top of the items I_{21} and I_{31}, i.e. $\gamma_{11} = \gamma_{21} + h_{21}$ and $\gamma_{11} = \gamma_{31} + h_{31}$. The respective areas they support are marked by the striped areas.

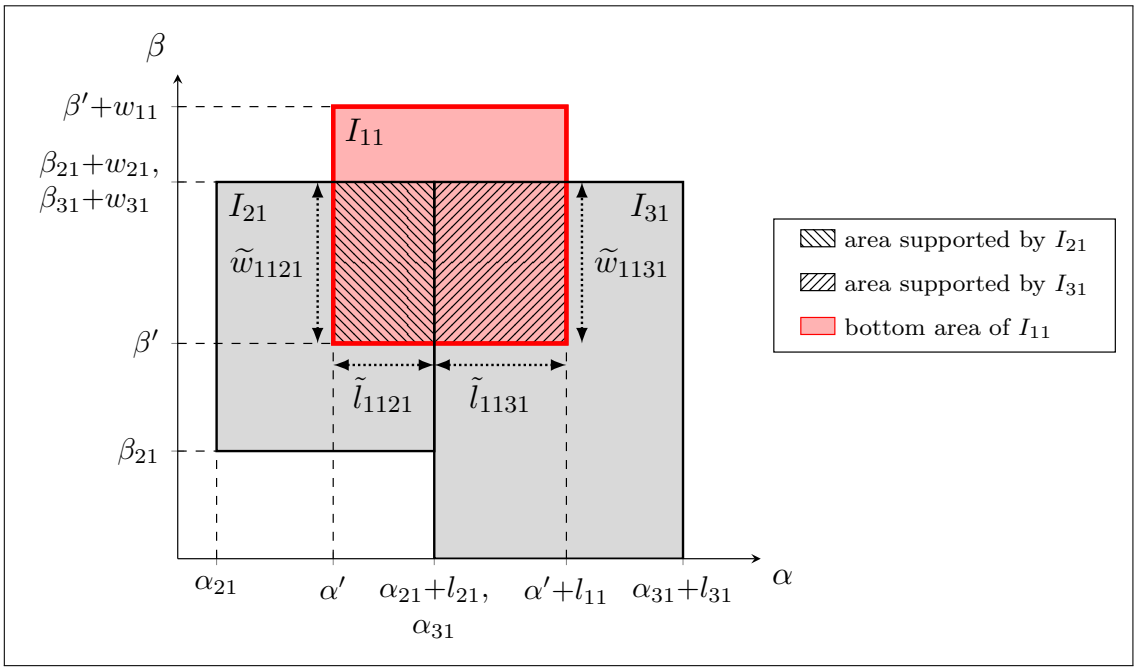


Figure 2.27: Determination of support areas provided for vertical stability

The right-hand side of (2.59) equals $VSP \cdot l_{ik} \cdot w_{ik}$ if item I_{ik} is placed in vehicle v with its back-left-bottom corner point in $(\alpha', \beta', \gamma')$ and is delivered in stage t. This

is the area of its bottom face that must at least be supported by other items. The constraint is not formulated for $\gamma' = 0$, i.e. items that are placed on the loading space floor do not need to satisfy a vertical stability constraint.

On the left-hand side of (2.59), the supporting area provided by items placed directly below the reference item I_{ik} is added up. In order to determine those items, the relations (2.57) and (2.58) are employed for the summation indices of α and β.

Note that only those constellations of items are considered, where the supporting item is delivered at the same stage or later than the item on top. This aspect is not strictly necessary in formulation (2.59). The fact that an item is exclusively supported (directly or indirectly) by items that are unloaded later or at the same stage is ensured by the LIFO constraint (2.70) (see below). Thus, its consideration in (2.59) merely serves to reduce the summations.

The previous formulation is adopted from Junqueira et al. (2012a,b). In the following, a modified formulation is presented, which requires the introduction of some new variables.[12] The vertical stability constraint can also be formulated using the occupation variables $\varphi^{ikv}_{\alpha\beta\gamma}$ introduced above (see (2.54)):

$$\begin{aligned} &\sum_{j\in N_c}\sum_{q\in J_j}\sum_{\left\{\alpha\in A_0\,\middle|\,\alpha'\le\alpha\le\alpha'+\widehat{l}_{ik}-1\right\}}\sum_{\left\{\beta\in B_0\,\middle|\,\beta'\le\beta\le\beta'+\widehat{w}_{ik}-1\right\}}\varphi^{jqv}_{\alpha\beta(\gamma'-1)} \\ &\ge VSP\cdot l_{ik}\cdot w_{ik}\cdot\pi^{iktv}_{\alpha'\beta'\gamma'} \end{aligned} \tag{2.62}$$

$$\forall\, i\in N_c, k\in J_i, t\in N\setminus\{n\}, v\in V, \alpha'\in A_{ik}, \beta'\in B_{ik}, \gamma'\in \Gamma_{ik}\setminus\{0\}.$$

That is, the number of occupied points directly below the base area of item I_{ik} is summed on the left-hand side (at height level $\gamma'-1$ if I_{ik} is placed at γ'). The number of occupied points must be greater than or equal to the minimum supporting area $VSP\cdot l_{ik}\cdot w_{ik}$. This formulation is applicable as all item dimensions and placement positions are assumed to be integer (see above).

Furthermore, let $supp_{ikjq}$ be the base area of item I_{ik} that is directly supported by another item I_{jq}. In Figure 2.27, it would refer to the individual striped areas. It can be determined as follows:

[12] As before, this might not be necessary at this point yet, but the variables will be required later and could then be used to propose a reformulation of the constraints presented here. In order to avoid "jumping" back to this part, the variables and the reformulated constraints are presented here.

$$
\begin{aligned}
supp_{ikjq} = & \sum_{v \in V} \sum_{\alpha' \in A_{ik}} \sum_{\beta' \in B_{ik}} \sum_{\gamma' \in \Gamma_{ik}} \left(\sum_{t \in N \setminus \{n\}} \pi^{iktv}_{\alpha'\beta'\gamma'} \right. \\
& \left. \cdot \sum_{\left\{\alpha \in A_0 \middle| \alpha' \leq \alpha \leq \alpha' + \widehat{l}_{ik} - 1\right\}} \sum_{\left\{\beta \in B_0 \middle| \beta' \leq \beta \leq \beta' + \widehat{w}_{ik} - 1\right\}} \varphi^{jqv}_{\alpha\beta(\gamma'-1)} \right) \\
& \forall\, i, j \in N_c, k \in J_i, q \in J_j.
\end{aligned} \tag{2.63}
$$

Pairs of items transported by the same vehicle are considered. The formula adds up all the points that are occupied by I_{jq} in the rectangular area given by the points $(\alpha', \beta', \gamma' - 1)$ and $(\alpha' + \widehat{l}_{ik} - 1, \beta' + \widehat{w}_{ik} - 1, \gamma' - 1)$ if I_{ik} is placed with its back-left-bottom corner point at $(\alpha', \beta', \gamma')$. Alternatively, it can be calculated as:

$$
\begin{aligned}
supp_{ikjq} = & \sum_{t \in N \setminus \{n\}} \sum_{v \in V} \sum_{\alpha' \in A_{ik}} \sum_{\beta' \in B_{ik}} \sum_{\gamma' \in \Gamma_{ik}} \left(\pi^{iktv}_{\alpha'\beta'\gamma'} \right. \\
& \cdot \sum_{\{u \in N | t \leq u < n\}} \sum_{\left\{\alpha \in A_{jq} | \alpha' - \widehat{l}_{jq} + 1 \leq \alpha \leq \alpha' + \widehat{l}_{ik} - 1\right\}} \\
& \left. \sum_{\{\beta \in B_{jq} | \beta' - \widehat{w}_{jq} + 1 \leq \beta \leq \beta' + \widehat{w}_{ik} - 1\}} \tilde{l}_{ikjq} \cdot \widetilde{w}_{ikjq} \cdot \pi^{jquv}_{\alpha\beta(\gamma' - h_{jq})} \right)
\end{aligned} \tag{2.64}
$$

where

$$
\tilde{l}_{ikjq} = \min(\alpha + \widehat{l}_{jq}, \alpha' + \widehat{l}_{ik}) - \max(\alpha, \alpha') \text{ and} \tag{2.65}
$$

$$
\widetilde{w}_{ikjq} = \min(\beta + \widehat{w}_{jq}, \beta' + \widehat{w}_{ik}) - \max(\beta, \beta') \tag{2.66}
$$

$$
\forall\, i, j \in N_c, k \in J_i, q \in J_j.
$$

As before, only those items I_{jq} are considered that are delivered at the same stage as I_{ik} or at a later stage than I_{ik}.

Moreover, let a_{ik} be the supported base area of item I_{ik}:

$$
a_{ik} = l_{ik} \cdot w_{ik} \cdot \sum_{\alpha \in A_{ik}} \sum_{\beta \in B_{ik}} \sum_{t \in N \setminus \{n\}} \sum_{v \in V} \pi^{iktv}_{\alpha\beta 0} + \sum_{j \in N_c} \sum_{q \in J_j} supp_{ikjq} \tag{2.67}
$$

$$
\forall\, i \in N_c, k \in J_i.
$$

If I_{ik} is placed on the loading space floor, the summation in the first summand

on the right-hand side equals 1 and, consequently, a_{ik} equals the total base area of I_{ik}. Otherwise, a_{ik} equals the supporting area provided by the item(s) directly supporting I_{ik} (second summand).
With the help of these variables, the vertical stability constraint (2.59) can be reformulated as:

$$a_{ik} \geq VSP \cdot l_{ik} \cdot w_{ik} \qquad \forall\ i \in N_c, k \in J_i. \tag{2.68}$$

For constraints formulated below it is important that $a_{ik} > 0$, which is ensured by (2.68) and $VSP > 0$ (see p. 12.)

Fragility (P6)
The (qualitative) fragility constraint requires that no non-fragile item ($f_{ik} = 0$) is placed on top of a fragile item ($f_{ik} = 1$), but fragile items can be stacked above each other. The constraint can be formulated as:

$$\begin{aligned} &\sum_{j \in N_c} \sum_{q \in J_j} \sum_{\{u \in N | t \leq u < n\}} \\ &\sum_{\left\{\alpha \in A_{jq} \middle| \alpha' - \widehat{l}_{jq} + 1 \leq \alpha \leq \alpha' + \widehat{l}_{ik} - 1\right\}} \\ &\sum_{\left\{\beta \in B_{jq} \middle| \beta' - \widehat{w}_{jq} + 1 \leq \beta \leq \beta' + \widehat{w}_{ik} - 1\right\}} f_{jq} \cdot \pi^{jquv}_{\alpha\beta(\gamma' - h_{jq})} \\ &\leq \left(1 - (1 - f_{ik}) \cdot \pi^{iktv}_{\alpha'\beta'\gamma'}\right) \cdot M_2 \end{aligned} \tag{2.69}$$

$$\forall\ i \in N_c, k \in I_i, t \in N \setminus \{n\}, v \in V, \alpha' \in A_{ik}, \beta' \in B_{ik}, \gamma' \in \Gamma_{ik} \setminus \{0\}.$$

If the reference item I_{ik} is placed with its corner point at $(\alpha', \beta', \gamma')$, the number of fragile items placed *directly* beneath the reference item I_{ik}, i.e. the items whose top faces touch the bottom face of I_{ik}, is summed on the left-hand side of (2.69). If I_{ik} is fragile, the right-hand side of the inequality equals M_2, which is a sufficiently large number (e.g. $M_2 \geq \sum_{i \in N_c} m_i$). Thus, any item can be placed beneath it. If I_{ik} is not fragile, the right-hand side equals zero. In order to fulfil the inequality, no fragile item can be placed directly below I_{ik}.

LIFO (P7a)
The LIFO constraint – stating that no item must be placed above or in front of

another item that is delivered earlier – can be formulated as follows:

$$\begin{aligned}
&\sum_{j\in N_c}\sum_{q\in J_j}\sum_{\{u\in N|t<u<n\}}\sum_{\left\{\alpha\in A_{jq}\middle|\alpha'-\widehat{l}_{jq}+1\le\alpha\le\alpha'+\widehat{l}_{ik}-1\right\}}\\
&\sum_{\left\{\beta\in B_{jq}\middle|\beta'-\widehat{w}_{jq}+1\le\beta\le\beta'+\widehat{w}_{ik}-1\right\}}\sum_{\left\{\gamma\in\Gamma_{jq}\middle|\gamma\ge\gamma'+h_{ik}\right\}}\pi^{jquv}_{\alpha\beta\gamma} \qquad (2.70)\\
&\le\left(1-\pi^{iktv}_{\alpha'\beta'\gamma'}\right)\cdot M_3\\
&\forall\, i\in N_c, k\in J_i, t\in N\setminus\{n\}, v\in V, \alpha'\in A_{ik}, \beta'\in B_{ik}, \gamma'\in\Gamma_{ik},
\end{aligned}$$

and

$$\begin{aligned}
&\sum_{j\in N_c}\sum_{q\in J_j}\sum_{\{u\in N|t<u<n\}}\sum_{\left\{\alpha\in A_{jq}\middle|\alpha\ge\alpha'+\widehat{l}_{ik}\right\}}\\
&\sum_{\left\{\beta\in B_{jq}\middle|\beta'-\widehat{w}_{jq}+1\le\beta\le\beta'+\widehat{w}_{ik}-1\right\}}\sum_{\left\{\gamma\in\Gamma_{jq}\middle|\gamma'-h_{jq}+1\le\gamma\le\gamma'+h_{ik}-1\right\}}\pi^{jquv}_{\alpha\beta\gamma} \qquad (2.71)\\
&\le\left(1-\pi^{iktv}_{\alpha'\beta'\gamma'}\right)\cdot M_3\\
&\forall\, i\in N_c, k\in J_i, t\in N\setminus\{n\}, v\in V, \alpha'\in A_{ik}, \beta'\in B_{ik}, \gamma'\in\Gamma_{ik}.
\end{aligned}$$

The first formulation (2.70) refers to the LIFO constraint along the γ-axis, i.e. preventing items from being placed above items that are delivered earlier. The right-hand side equals zero if the point $(\alpha', \beta', \gamma')$ in vehicle v is the back-left-bottom corner point of an item I_{ik} delivered in stage t. Otherwise, it assumes a sufficiently large value M_3 (e.g. $M_3 \geq \sum_{i\in N_c} m_i$). On the left-hand side of the constraint, the number of items that are placed above item I_{ik} and are delivered at a later stage ($u > t$) is summed. That is, if item I_{ik} is placed at $(\alpha', \beta', \gamma')$ in vehicle v and delivered in stage t, the number of items placed above it and delivered at any stage $u > t$ must equal zero. Note that not only those items that are stacked *directly* on top of each other are considered, but all items that are somewhere above item I_{ik} must be taken into account (see Chapter 2.1.1).
Analogously, (2.71) refers to the LIFO constraint along the α-axis, i.e. preventing items from being placed in front of items that are delivered earlier.

Reachability (P8)

In order to enable the operator (e.g. human worker or forklift) to unload the items without further rearrangement or having to step onto other items, it must be ensured

that an item is reachable. For this purpose, the parameter λ is used, which represents the operator's maximum horizontal reach along the α-axis. Let δ_{ik} $(i \in N_c, k \in J_i)$ be the closest possible position along the α-axis of the operator when unloading item I_{ik}. $\alpha_{ik} + \widehat{l}_{ik}$ represents the α-coordinate of the front side (towards the unloading side) of item I_{ik}. The reachability constraint can be stated as:

$$\delta_{ik} - (\alpha_{ik} + \widehat{l}_{ik}) \leq \lambda \qquad \forall\ i \in N_c, k \in J_i. \tag{2.72}$$

The position δ_{ik} is determined as the maximum of the α-coordinate of the front side of item I_{ik} $(\alpha_{ik} + \widehat{l}_{ik})$ and the α-coordinates of the front sides of all other relevant items. The relevant items to be considered are those items that are transported in the same vehicle and would block the unloading of I_{ik}. This includes items that are either delivered at a later stage in the route or at the same stage, and are placed below item I_{ik} (cf. Figure 2.12). Figure 2.28 illustrates items that do *not* need to be considered. The top view of three items, which are placed next to each other, is shown. The customers are visited in the sequence $(1, 2)$. Although the items I_{21} and I_{22} are longer than item I_{11}, they do not block the way to it. Thus, the operator can step up until item I_{11}.

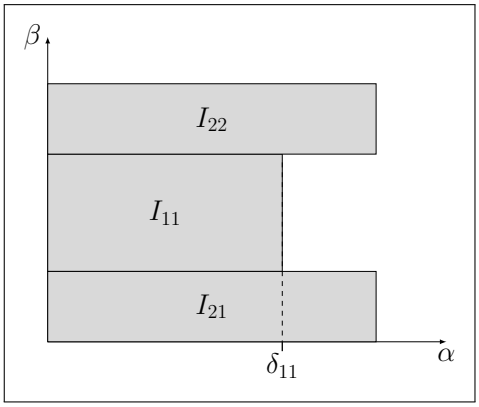


Figure 2.28: Item arrangement and operator position δ_{11} for an item I_{11}

In conclusion, position δ_{ik} equals the maximum of

- the front edge of item I_{ik},
- the maximum front edge of the items I_{jq} $(j \neq i)$ that are unloaded after I_{ik} and are placed so that $\beta_{ik} - \widehat{w}_{jq} + 1 \leq \beta_{jq} \leq \beta_{ik} + \widehat{w}_{ik} - 1$,
- the maximum front edge of the remaining items of customer i that are placed below I_{ik}.

The position δ_{ik} can be determined as:

$$\delta_{ik} = \sum_{t \in N \setminus \{n\}} \sum_{v \in V} \left(\sum_{\alpha \in A_{ik}} \sum_{\beta \in B_{ik}} \sum_{\gamma \in \Gamma_{ik}} \left(\pi^{iktv}_{\alpha\beta\gamma} \cdot \max \left(\alpha + \widehat{l}_{ik}, \right. \right. \right.$$

$$\max_{\substack{j \in N_c \setminus \{i\}, \\ q \in J_j}} \left(\sum_{\{u \in N \mid t < u < n\}} \sum_{\left\{ \alpha' \in A_{jq} \middle| \alpha' \geq \alpha + \widehat{l}_{ik} - \widehat{l}_{jq} + 1 \right\}} \right.$$

$$\left. \left. \sum_{\left\{ \beta' \in B_{ik} \middle| \beta - \widehat{w}_{jq} + 1 \leq \beta' \leq \beta + \widehat{w}_{ik} - 1 \right\}} \sum_{\left\{ \gamma' \in \Gamma_{jq} \middle| \gamma' \leq \gamma - h_{jq} \right\}} \pi^{jquv}_{\alpha'\beta'\gamma'} \cdot \left(\alpha' + \widehat{l}_{jq} \right) \right), \right. \tag{2.73}$$

$$\max_{q \in J_i \setminus \{k\}} \left(\sum_{\left\{ \alpha' \in A_{iq} \middle| \alpha + \widehat{l}_{ik} - \widehat{l}_{iq} + 1 \leq \alpha' \leq \alpha + \widehat{l}_{ik} - 1 \right\}} \sum_{\left\{ \beta' \in B_{iq} \middle| \beta - \widehat{w}_{iq} + 1 \leq \beta' \leq \beta + \widehat{w}_{ik} - 1 \right\}} \right.$$

$$\left. \left. \left. \left. \sum_{\left\{ \gamma' \in \Gamma_{iq} \middle| \gamma' \leq \gamma - h_{iq} \right\}} \pi^{iqtv}_{\alpha'\beta'\gamma'} \cdot \left(\alpha' + \widehat{l}_{iq} \right) \right) \right) \right) \right) \right) .$$

The outer summations fix the stage and vehicle indices t and v so that only those other items are considered that are also transported by vehicle v and are delivered later than the reference item I_{ik} (if they are of different customers). By means of the following three summations, the respective position of I_{ik} is fixed for the next bracket. Thus, the point (α, β, γ) is the position of the back-left-bottom corner point of I_{ik} in the expressions that follow. The first term in the outer max-expression $(\alpha + \widehat{l}_{ik})$ corresponds to the front edge of I_{ik}. The second term refers to the front edges of the items of other customers that are visited after customer i. Only those items are considered, whose front edges are at larger α-coordinates than the front edge of I_{ik}. The front edge of item I_{jq} with its back edge at α_{jq} is at a larger α-coordinate than the front edge of I_{ik} with its back at α_{ik} if:

$$\alpha_{jq} \geq \alpha_{ik} + \widehat{l}_{ik} - \widehat{l}_{jq} + 1. \tag{2.74}$$

In addition, the term considers only those items that overlap with I_{ik} on the β-axis (see above) and are placed below I_{ik}. The latter is also ensured by the LIFO

constraint and serves to reduce the summation. Finally, the third term corresponds to the maximum front edge of other items of customer i that are placed below I_{ik}. The constraint is illustrated in Figure 2.29. The customers corresponding to the six depicted items are visited in the sequence (1, 2, 3, 4, 5, 6). When unloading the items I_{11} and I_{21}, the operator has the illustrated position $\delta_{11} = \delta_{21}$. Thus, item I_{11} is reachable and can be unloaded since its distance to the operator $(\delta_{11} - (\alpha_{11} + \widehat{l}_{11}))$ is shorter than λ (shaded area). Item I_{21}, however, is too far away. The same applies to item I_{41}. When this item is unloaded, the operator can step up to item I_{61} (δ_{41}), as the items I_{11}, I_{21} and I_{31} have already been unloaded. However, the distance to I_{41} $(\delta_{41} - (\alpha_{41} + \widehat{l}_{41}))$ is still too large.

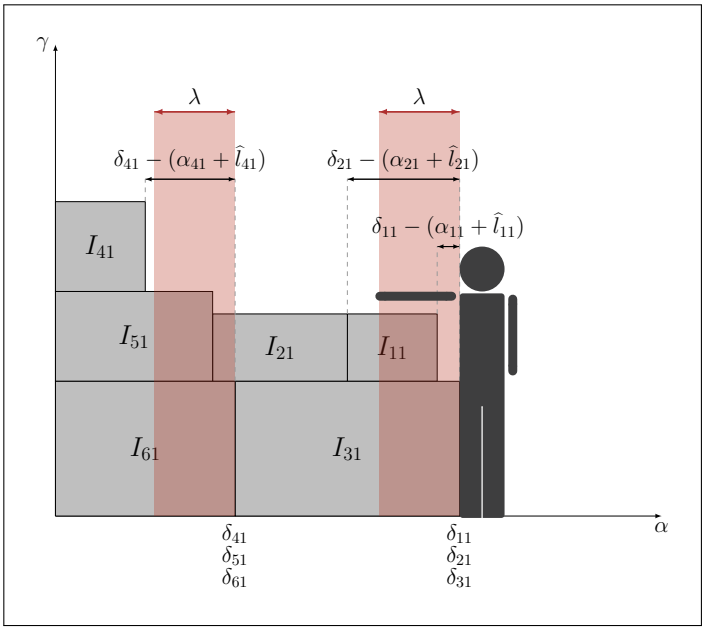


Figure 2.29: Item arrangement with relevant variables for the evaluation of reachability

If there is no item that would block the way to unload an item I_{ik}, the position δ_{ik} equals the position of the front side of item I_{ik}. In Figure 2.29, this would be the case for the items I_{31} and I_{61}.

Robust stability (P9)

The vertical stability constraint presented above is extended in order to ensure robust stability, i.e. the required supporting area must not only be provided by the items that are directly supporting the reference item, but by all items placed below it.

An arrangement is illustrated in Figure 2.30. It is assumed that all items have the same width and their corner points have the same β-coordinates. Thus, the

supporting areas depend only on the lengths of the items. $\tilde{l}_{ikjq}$ represents the length of the part of the bottom area of item I_{ik} that is supported by item I_{jq}, i.e.

$$\tilde{l}_{ikjq} = \min\left(\alpha_{ik} + \widehat{l}_{ik}, \alpha_{jq} + \widehat{l}_{jq}\right) - \max\left(\alpha_{ik}, \alpha_{jq}\right). \tag{2.75}$$

In the example (Figure 2.30), $\tilde{l}_{2131}$ is assumed to be sufficient to support item I_{21}, whereas $\tilde{l}_{2141} + \tilde{l}_{2151}$ as well as $\tilde{l}_{2161}$ are too short. Note, that item I_{71} must not be considered here. Although item I_{21} is placed above item I_{71}, I_{71} does not offer any support for I_{21}.

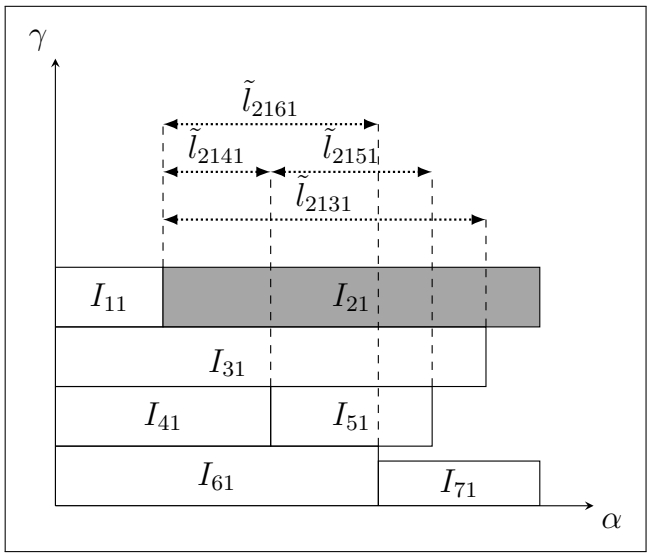


Figure 2.30: Illustration of the robust stability constraint (Adapted from Ceschia et al., 2013)

In order to formulate the robust stability constraint, new variables are introduced: The variables $\sigma^v_{\alpha\beta\gamma}$ ($v \in V, \alpha \in A_0, \beta \in B_0, \gamma \in \Gamma_0$) describe whether a point (α, β, γ) in vehicle v is fully supported, i.e. whether *all* points below it are occupied by any item:

$$\sigma^v_{\alpha\beta\gamma} = \begin{cases} 1, & \text{if } \sum_{i\in N_c}\sum_{k\in J_i}\sum_{\{\gamma'\in\Gamma_{ik}|\gamma'<\gamma\}} \varphi^{ikv}_{\alpha\beta\gamma'} = \gamma, \text{ or } \gamma = 0, \\ 0, & \text{otherwise.} \end{cases} \tag{2.76}$$

The application of these variables is illustrated in Figure 2.31 where the grey item is the reference item that should be tested for robust stability. The points it occupies on its bottom face are depicted by the black and white dots. The black ones are fully supported, i.e. all points below them are occupied (green dots). However, the white ones are not (fully) supported as some points below them are unoccupied

(crosses). Hence, twelve of the 21 points on the bottom face are fully supported, which corresponds to a robust stability factor of about 57 %.

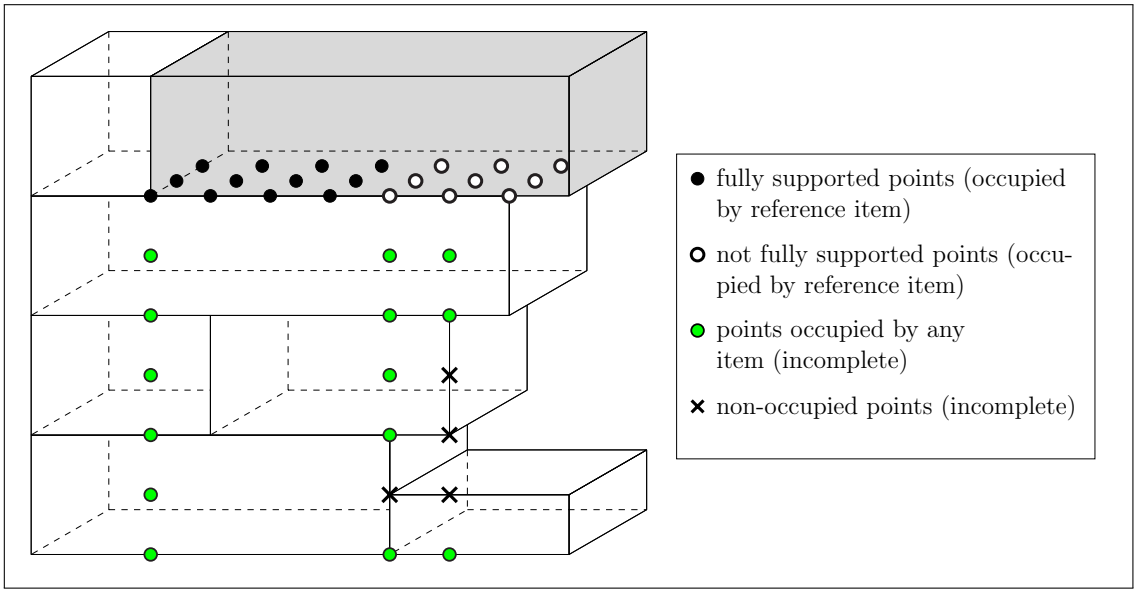


Figure 2.31: Occupied and supported points

Using the variables defined above, the robust stability constraint can be formulated as:

$$\sum_{\left\{\alpha\in A_0 \middle| \alpha'\leq\alpha\leq\alpha'+\widehat{l}_{ik}-1\right\}} \sum_{\left\{\beta\in B_0 \middle| \beta'\leq\beta\leq\beta'+\widehat{w}_{ik}-1\right\}} \sigma^{v}_{\alpha\beta\gamma'} \geq VSP \cdot l_{ik} \cdot w_{ik} \cdot \sum_{t\in N\setminus\{n\}} \pi^{iktv}_{\alpha'\beta'\gamma'} \tag{2.77}$$

$$\forall\ i \in N_c, k \in J_i, v \in V, \alpha' \in A_{ik}, \beta' \in B_{ik}, \gamma' \in \Gamma_{ik} \setminus \{0\}.$$

If item I_{ik} is placed in vehicle v with its back-left-bottom corner point at $(\alpha', \beta', \gamma')$ (with $\gamma' > 0$), the term on the right-hand side equals the minimum number of points on the bottom face of I_{ik} that need to be fully supported in order to satisfy the constraint. Otherwise, it equals 0. On the left-hand side, the number of fully supported points in the rectangular area given by the points $(\alpha', \beta', \gamma')$ and $(\alpha' + \widehat{l}_{ik} - 1, \beta' + \widehat{w}_{ik} - 1, \gamma')$ is summed.

This formulation does not contain any information about the stage in which the items are unloaded. As mentioned above, this aspect is not strictly necessary as the LIFO constraint ensures that the supporting items are not delivered earlier than the reference item.

Load bearing strength (P10)

Every item I_{ik} has a weight d_{ik} and is assigned a load bearing strength p_{ik}, which

is the maximum weight (pressure) it can bear on any unit area of its top face and which must not be exceeded by the weights of the items stacked on top of I_{ik}.
In order to model the load bearing strength constraint, some assumptions have to be made. If this kind of restriction is considered in the CLP and 3L-VRP literature, the assumptions proposed by Ratcliff and Bischoff (1998) are usually applied:

> "It is assumed that the weight of an item is spread in direct proportion to the contact areas with the boxes on which it rests - *and that it acts straight downwards rather than being distributed over the whole of a supporting face.*" (Ratcliff and Bischoff, 1998, p. 66)

Moreover, Ratcliff and Bischoff (1998) assume that an item's load bearing strength is equal for all unit areas of its top face (although a box might be able to bear a greater weight along its edges than in the centre of the face) and all items are fully supported.
The transmission of the weight of stacked items according to Ratcliff and Bischoff (1998) is illustrated in Figure 2.32. The arrows represent the directions of weight transmissions. The weight of item I_{11} (blue) is only loaded on the areas directly beneath it. Item I_{21} (yellow), for example, bears the weight of I_{11} only on one fourth of its top face, and it only transmits the (partial) weight of I_{11} onto the top face areas of other items that are beneath I_{11} (overlapping α-β-projection). Hence, although the lower left item (I_{41}) carries I_{21}, no weight of item I_{11} is transmitted to it.

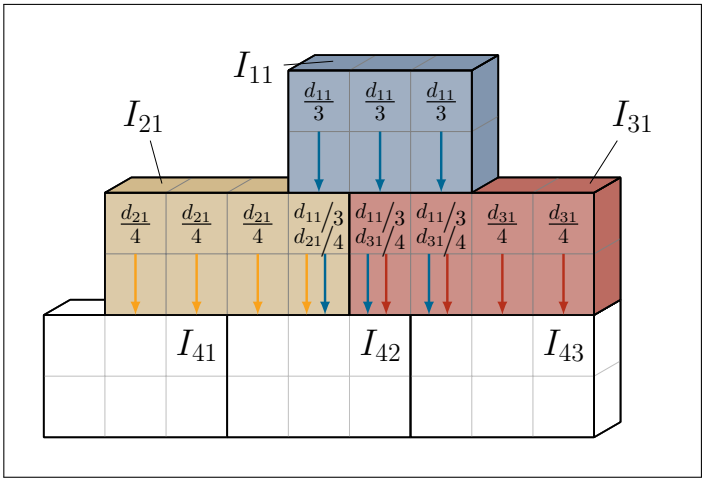


Figure 2.32: Load transmission according to Ratcliff and Bischoff (1998)

Under the presented assumptions, the formulation of the load bearing strength constraint for the CLP proposed by (Junqueira et al., 2012b) can be adapted to the

3L-VRPTW as follows:

$$\begin{aligned}
&\sum_{j\in N_c}\sum_{q\in J_j}\sum_{u\in N\setminus\{n\}} \\
&\quad\sum_{\left\{\alpha\in A_{jq}\middle|\alpha'-\widehat{l}_{jq}+1\le\alpha\le\alpha'\right\}}\ \sum_{\left\{\beta\in B_{jq}\middle|\beta'-\widehat{w}_{jq}+1\le\beta\le\beta'\right\}}\ \sum_{\left\{\gamma\in\Gamma_{jq}\middle|\gamma'+1\le\gamma\le H-h_{jq}\right\}} \left(\frac{d_{jq}}{l_{jq}\cdot w_{jq}}\right)\cdot\pi_{\alpha\beta\gamma}^{jquv} \\
&\le\sum_{i\in N_c}\sum_{k\in J_i}\sum_{t\in N\setminus\{n\}} \\
&\quad\sum_{\left\{\alpha\in A_{ik}\middle|\alpha'-\widehat{l}_{ik}+1\le\alpha\le\alpha'\right\}}\ \sum_{\left\{\beta\in B_{ik}\middle|\beta'-\widehat{w}_{ik}+1\le\beta\le\beta'\right\}}\ \sum_{\left\{\gamma\in\Gamma_{ik}\middle|\gamma'-h_{ik}+1\le\gamma\le\gamma'\right\}} p_{ik}\cdot\pi_{\alpha\beta\gamma}^{iktv}
\end{aligned} \tag{2.78}$$

$$\forall\ v\in V,\alpha'\in A_0,\beta'\in B_0,\gamma'\in\Gamma_0.$$

That is, if point (α',β',γ') in vehicle v is occupied, the right-hand side of (2.78) equals the load bearing strength p_{ik} of the item I_{ik} that occupies the point. On the left-hand side, the proportional weights of all items occupying the points (α',β',γ) with $\gamma>\gamma'$ and whose bottom area is at $\gamma>\gamma'$, is added up. An item I_{jq} loads a weight of $\frac{d_{jq}}{l_{jq}\cdot w_{jq}}$ on each point (or unit area) below it. The fact that the bottom face of the upper items must be at $\gamma>\gamma'$ ensures that the lower item (whose load bearing strength is determined on the right-hand side) is not loaded by its own weight as the items regarded on the left-hand side must have their bottom face above γ'. Although the lower item can occupy further points above (α',β',γ'), its bottom face is at a height level $\gamma\le\gamma'$.

The approach of Ratcliff and Bischoff (1998) is a simplification of the actual physical settings. In the following, a more realistic formulation is attempted. The assumption is still made that the load bearing strength is equal over the whole top face of an item. Furthermore, the formulation should also allow the consideration of overhanging items due to $VSP<1$. The weight transmitted via one unit area of the bottom face of an item is, therefore, not based on the whole base area of that item, but on the base area that is actually supported by other items (cf. e.g. Christensen and Rousøe, 2009). In the example depicted in Figure 2.33, each unit area of the top face of the lower item bears one third of the upper item's weight, because only three of four unit areas of its base area are supported. Thus, a formulation of the load bearing strength is required that distributes the whole weight of an item over its actually supported bottom area. The formulation (2.78) does not ensure that.

According to it, one fourth of the weight would be transmitted per base unit area as it does not take less-than-full support into account.

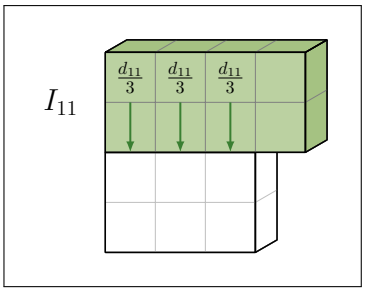


Figure 2.33: Load transmission of overhanging items

For the formulation of the load bearing strength constraint, the supporting variables a_{jq} and $supp_{jqik}$ $(i, j \in N_c, k \in J_i, q \in J_j)$ are used in the following. $supp_{jqik}$ denotes the base area of I_{jq} that is supported by I_{ik} (see (2.63), (2.64)). a_{jq} refers to the base area of I_{jq} that is supported by any item or the container floor (see (2.67)). Thus, the (own) weight transmitted by an item I_{jq} to any item below it corresponds to $\frac{d_{jq}}{a_{jq}}$.

Furthermore, let $\widehat{d}_{ik}$ $(i \in N_c, k \in J_i)$ be the weight of item I_{ik} including the weight transmitted to it from above:

$$\widehat{d}_{ik} = d_{ik} + \sum_{j \in N_c} \sum_{q \in J_j} \left(\frac{supp_{jqik}}{a_{jq}} \cdot \widehat{d}_{jq} \right). \tag{2.79}$$

$\frac{supp_{jqik}}{a_{jq}}$ is the proportion of the supported base area of I_{jq} that is supported by I_{ik}. The reference item I_{ik} bears $\frac{supp_{jqik}}{a_{jq}} \cdot 100$ % of the weight of any item I_{jq} that is placed directly on top of it.

If all items can be seen as rigid bodies, i.e. they are inelastic, it can be assumed that the transmitted weight is spread evenly over the (supported) base area of an item. That is, on each unit area of its bottom face, an item transmits the same weight downwards, regardless of whether another item (whose weight is also transmitted) is placed above the respective unit area or not. Figure 2.34 shows the same arrangement as in Figure 2.32, but with the load transmission according to the mentioned assumptions. Item I_{21} spreads its own weight (yellow arrows) and the weight it transmits from item I_{11} (blue arrows) evenly to all top faces of the other items it is directly placed above. The same accounts for item I_{31}.

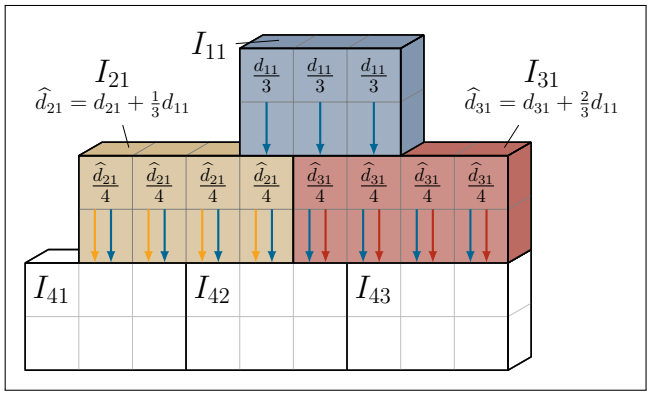


Figure 2.34: Complete load transmission

For the mathematical model, the load bearing strength constraint can be stated as:

$$\begin{aligned}
&\sum_{j\in N_c}\sum_{q\in J_j}\sum_{\{u\in N|t\le u<n\}}\sum_{\left\{\alpha\in A_{jq}\middle|\alpha'-\widehat{l}_{jq}+1\le\alpha\le\alpha'\right\}}\sum_{\left\{\beta\in B_{jq}\middle|\beta'-\widehat{w}_{jq}+1\le\beta\le\beta'\right\}}\frac{\widehat{d}_{jq}}{a_{jq}}\cdot\pi^{jquv}_{\alpha\beta(\gamma'+h_{ik})}\\
&\le\sum_{\left\{\alpha\in A_{ik}\middle|\alpha'-\widehat{l}_{ik}+1\le\alpha\le\alpha'\right\}}\sum_{\left\{\beta\in B_{ik}\middle|\beta'-\widehat{w}_{ik}+1\le\beta\le\beta'\right\}}p_{ik}\cdot\pi^{iktv}_{\alpha\beta\gamma'}\\
&+\left(1-\sum_{\left\{\alpha\in A_{ik}\middle|\alpha'-\widehat{l}_{ik}+1\le\alpha\le\alpha'\right\}}\sum_{\left\{\beta\in B_{ik}\middle|\beta'-\widehat{w}_{ik}+1\le\beta\le\beta'\right\}}\pi^{iktv}_{\alpha\beta\gamma'}\right)\cdot M_4
\end{aligned}\tag{2.80}$$

$$\forall\, i\in N_c, k\in J_i, v\in V, t\in N\setminus\{n\}, \alpha'\in A_0, \beta'\in B_0, \gamma'\in \Gamma_0.$$

The right-hand side equals p_{ik} if $(\alpha', \beta', \gamma')$ in vehicle v is occupied by item I_{ik} that is delivered in stage t, and if the γ-coordinate of its corner point is at height γ'. Otherwise, the right-hand side equals a sufficiently large value M_4 (e.g. $M_4 \ge \sum_{i\in N_c}\sum_{k\in J_i} d_{ik}$ weight units per unit area). That is, any pressure can be applied to the corresponding unit area. This is a further addition to (2.78). The left-hand side corresponds to the weight transmitted by the item placed directly on top of I_{ik} (γ-coordinate of its corner point is at $\gamma'+h_{ik}$) and occupying the point $(\alpha', \beta', \gamma'+h_{ik})$.

Variable domain constraints

(2.81) to (2.83) represent the domain constraints of the decision variables:

$$x^{tv}_{ij}\in\{0,1\} \qquad \forall\, i,j,t\in N, v\in V. \tag{2.81}$$

$$\pi^{iktv}_{\alpha\beta\gamma}\in\{0,1\} \qquad \forall\, i\in N_c, k\in J_i, t\in N\setminus\{n\}, v\in V, \quad \alpha\in A_{ik}, \beta\in B_{ik}, \gamma\in\Gamma_{ik}, \tag{2.82}$$

$$o_{ik} \in \{0,1\} \qquad \forall\, i \in N_c, k \in J_i, \tag{2.83}$$

(2.84) to (2.93) represent the domain constraints of the variables depending on the decision variables. Here, the defining restrictions or equations usually determine the respective domains.

$$\widehat{l}_{ik}, \widehat{w}_{ik} \in \{l_{ik}, w_{ik}\} \qquad \forall\, i \in N_c, k \in J_i, \tag{2.84}$$

$$\alpha_{ik} \in A_{ik} \qquad \forall\, i \in N_c, k \in J_i, \tag{2.85}$$

$$\beta_{ik} \in B_{ik} \qquad \forall\, i \in N_c, k \in J_i, \tag{2.86}$$

$$\gamma_{ik} \in \Gamma_{ik} \qquad \forall\, i \in N_c, k \in J_i, \tag{2.87}$$

$$start_{iv} \in \mathbb{R}_0^+ \qquad \forall\, i \in N, v \in V, \tag{2.88}$$

$$\varphi_{\alpha\beta\gamma}^{ikv} \in \{0,1\} \qquad \forall\, i \in N_c, k \in J_i, v \in V, \tag{2.89}$$

$$\alpha \in A_0, \beta \in B_0, \gamma \in \Gamma_0,$$

$$supp_{ikjq} \in \mathbb{Z}_0^+ \qquad \forall\, i,j \in N_c, k \in J_i, q \in J_j, \tag{2.90}$$

$$a_{ik} \in [0, l_{ik} \cdot w_{ik}] \cap \mathbb{Z}_0^+ \qquad \forall\, i \in N_c, k \in J_i, \tag{2.91}$$

$$\widehat{d}_{ik} \in \mathbb{R}^+ \qquad \forall\, i \in N_c, k \in J_i, \tag{2.92}$$

$$\delta_{ik} \in [0, L] \qquad \forall\, i \in N_c, k \in J_i, \tag{2.93}$$

$$\sigma_{\alpha\beta\gamma}^{v} \in \{0,1\} \qquad \forall\, v \in V, \alpha \in A_0, \beta \in B_0, \gamma \in \Gamma_0. \tag{2.94}$$

2.2.2 The 3L-VRP with clustered backhauls and time windows

Sets, constants and decision variables

The same sets and constants as before can be applied here (see Table 2.3). In addition, n_L (n_B) denotes the number of linehaul (backhaul) customers contained in the set N^L (N^B). The same decision variables as in the previously presented model are utilized.

Model

In the following, the mathematical model for the 3L-VRPCB(TW) is presented. The explanations will not be as detailed as in the previous chapter, they will rather focus on the distinctions from the formulations in Chapter 2.2.1.

The objective function for the 3L-VRPCB(TW) is identical to the objective function (2.28) for the 3L-VRPTW.

Routing constraints

The routing constraints (2.29)-(2.34), (2.37) and (2.38), as well as the time window constraint (2.39)-(2.42) from the 3L-VRPTW can be applied unaltered to the 3L-VRPCB(TW).

In contrast to the 3L-VRPTW, each customer is either a linehaul or a backhaul customer in the 3L-VRPCB(TW). The particular characteristic of the problem is that in each route, all linehaul customers must be visited before the backhaul customers (R5). This is ensured by the following constraint:

$$x_{ij}^{tv} = 0 \qquad \forall\ i \in N^B, j \in N^L, t \in N, v \in V. \tag{2.95}$$

It prevents that a vehicle goes from a backhaul to a linehaul customer, i.e. once a backhaul customer is visited, the vehicle cannot approach any linehaul customers anymore. This formulation also allows tours consisting of backhaul customers only. Some formulations of the VRPCB demand that each tour must contain at least one linehaul customer. However, as this restriction is often omitted for the VRPCBTW (e.g. Ropke and Pisinger, 2006a), it is not considered here either.

As described in Chapter 2.1.2.2, the separation of linehaul and backhaul customers within a route allows for creating two separate packing patterns. Thus, most constraints can be formulated separately for linehauls and backhauls, and the formulations differ only slightly from those presented in Chapter 2.2.1.

The weight (2.35) and volume capacity constraints (2.36) can be re-formulated as:

(1) linehaul

$$\sum_{i \in N^L} \sum_{k \in J_i} \sum_{j \in N} \sum_{t \in N_c} d_{ik} \cdot x_{ij}^{tv} \leq D \qquad \forall\ v \in V, \tag{2.96}$$

$$\sum_{i \in N^L} \sum_{k \in J_i} \sum_{j \in N} \sum_{t \in N_c} l_{ik} \cdot w_{ik} \cdot h_{ik} \cdot x_{ij}^{tv} \leq L \cdot W \cdot H \qquad \forall\ v \in V; \tag{2.97}$$

(2) backhaul

$$\sum_{i \in N^B} \sum_{k \in J_i} \sum_{j \in N^B \cup \{0\}} \sum_{t \in N_c} d_{ik} \cdot x_{ij}^{tv} \leq D \qquad \forall\ v \in V, \tag{2.98}$$

$$\sum_{i\in N^B}\sum_{k\in J_i}\sum_{j\in N^B\cup\{0\}}\sum_{t\in N_c} l_{ik}\cdot w_{ik}\cdot h_{ik}\cdot x_{ij}^{tv} \le L\cdot W\cdot H \qquad \forall\, v\in V. \tag{2.99}$$

In the case of linehauls ((2.96) and (2.97)), the total weight and volume of the items in the vehicle are constantly decreasing until all items are delivered. In the case of backhauls ((2.98) and (2.99)), the total weight and volume are constantly increasing until all items are collected. That is, neither all linehaul items nor all backhaul items transported in the same tour must exceed the weight and volume capacity.

Packing constraints

Geometrical constraints (P1)-(P3)

The non-overlapping constraint (cf. (2.52)) must be stated separately for linehaul and backhaul customers, too, because otherwise a backhaul item could not take a linehaul item's place:

(1) linehaul:

$$\begin{aligned}&\sum_{i\in N^L}\sum_{k\in J_i}\sum_{t\in N\setminus\{n\}}\\ &\sum_{\left\{\alpha\in A_{ik}\mid \alpha'-\widehat{l}_{ik}+1\le\alpha\le\alpha'\right\}}\\ &\sum_{\{\beta\in B_{ik}\mid \beta'-\widehat{w}_{ik}+1\le\beta\le\beta'\}}\\ &\sum_{\{\gamma\in\Gamma_{ik}\mid \gamma'-h_{ik}+1\le\gamma\le\gamma'\}} \pi_{\alpha\beta\gamma}^{iktv}\le 1\\ &\forall\ \alpha'\in A_0,\beta'\in B_0,\gamma'\in\Gamma_0, v\in V;\end{aligned} \tag{2.100}$$

(2) backhaul:

$$\begin{aligned}&\sum_{i\in N^B}\sum_{k\in J_i}\sum_{t\in N\setminus\{n\}}\\ &\sum_{\left\{\alpha\in A_{ik}\mid \alpha'-\widehat{l}_{ik}+1\le\alpha\le\alpha'\right\}}\\ &\sum_{\{\beta\in B_{ik}\mid \beta'-\widehat{w}_{ik}+1\le\beta\le\beta'\}}\\ &\sum_{\{\gamma\in\Gamma_{ik}\mid \gamma'-h_{ik}+1\le\gamma\le\gamma'\}} \pi_{\alpha\beta\gamma}^{iktv}\le 1\\ &\forall\ \alpha'\in A_0,\beta'\in B_0,\gamma'\in\Gamma_0, v\in V.\end{aligned} \tag{2.101}$$

Alternatively, they can be formulated using the occupation variables $\varphi^{ikv}_{\alpha\beta\gamma}$:

$$\text{(1) linehaul:} \quad \sum_{i \in N^L} \sum_{k \in J_i} \varphi^{ikv}_{\alpha\beta\gamma} \leq 1 \quad \forall\, v \in V, \alpha \in A_0, \beta \in B_0, \gamma \in \Gamma_0; \tag{2.102}$$

$$\text{(2) backhaul:} \quad \sum_{i \in N^B} \sum_{k \in J_i} \varphi^{ikv}_{\alpha\beta\gamma} \leq 1 \quad \forall\, v \in V, \alpha \in A_0, \beta \in B_0, \gamma \in \Gamma_0. \tag{2.103}$$

The other geometrical constraint (2.45) – stating that an item must be placed completely within the loading space – can be applied unchanged.

Vertical stability (P5)

Considering the support variables, the vertical stability constraint can be stated as:

$$a_{ik} \geq VSP \cdot l_{ik} \cdot w_{ik} \qquad \forall\, i \in N_c, k \in J_i, \tag{2.104}$$

where

(1) linehaul:

$$a_{ik} = l_{ik} \cdot w_{ik} \cdot \sum_{\alpha \in A_{ik}} \sum_{\beta \in B_{ik}} \sum_{t \in N \setminus \{n\}} \sum_{v \in V} \pi^{iktv}_{\alpha\beta 0} + \sum_{j \in N^L} \sum_{q \in J_j} supp_{ikjq} \tag{2.105}$$

$$\forall\, i \in N^L, k \in J_i;$$

(2) backhaul:

$$a_{ik} = l_{ik} \cdot w_{ik} \cdot \sum_{\alpha \in A_{ik}} \sum_{\beta \in B_{ik}} \sum_{t \in N \setminus \{n\}} \sum_{v \in V} \pi^{iktv}_{\alpha\beta 0} + \sum_{j \in N^B} \sum_{q \in J_j} supp_{ikjq} \tag{2.106}$$

$$\forall\, i \in N^B, k \in J_i.$$

For the linehaul items, $supp_{ikjq}$ $(i, j \in N^L, k \in J_i, q \in J_j)$ can be determined as in (2.63) or (2.64). For the backhaul items, (2.63) can also be applied in order to determine $supp_{ikjq}$ $(i, j \in N^B, k \in J_i, q \in J_j)$. The alternative formulation (2.64) is modified by adjusting the domain for the stage index u of I_{jq}. Only those items need to be considered for the support that are collected earlier than I_{ik} (due to the LIFO constraint):

$$
\begin{aligned}
supp_{ikjq} = \sum_{t\in N\setminus\{n\}} \sum_{v\in V} \sum_{\alpha'\in A_{ik}} \sum_{\beta'\in B_{ik}} \sum_{\gamma'\in \Gamma_{ik}} \Bigg(& \pi^{iktv}_{\alpha'\beta'\gamma'} \\
& \cdot \sum_{\{u\in N|u\le t\}} \sum_{\left\{\alpha\in A_{jq}|\alpha'-\widehat{l}_{jq}+1\le\alpha\le\alpha'+\widehat{l}_{ik}-1\right\}} \\
& \sum_{\{\beta\in B_{jq}|\beta'-\widehat{w}_{jq}+1\le\beta\le\beta'+\widehat{w}_{ik}-1\}} \tilde{l}_{ikjq}\cdot\widetilde{w}_{ikjq}\cdot\pi^{jquv}_{\alpha\beta(\gamma'-h_{jq})}\Bigg).
\end{aligned}
\tag{2.107}
$$

Alternatively, the formulation (2.59) based on Junqueira et al. (2012a,b) can be applied by substituting the domains for the customers i and j with N^L or N^B, respectively, and by substituting the domains for the considered stages on the left-hand side of the inequations with $\{u \in N|u \le t\}$ in the case of backhaul customers.

Fragility (P6)

The fragility constraint for the 3L-VRPCB(TW) can be formulated as follows:

$$
\begin{aligned}
\text{(1) linehaul:}\quad & \sum_{j\in N^L}\sum_{q\in J_j}\sum_{\{u\in N|t\le u<n\}} \\
& \sum_{\left\{\alpha\in A_{jq}\middle|\alpha'-\widehat{l}_{jq}+1\le\alpha\le\alpha'+\widehat{l}_{ik}-1\right\}} \\
& \sum_{\left\{\beta\in B_{jq}\middle|\beta'-\widehat{w}_{jq}+1\le\beta\le\beta'+\widehat{w}_{ik}-1\right\}} f_{jq}\cdot\pi^{jquv}_{\alpha\beta(\gamma'-h_{jq})} \\
& \le \left(1-(1-f_{ik})\cdot\pi^{iktv}_{\alpha'\beta'\gamma'}\right)\cdot M_2
\end{aligned}
\tag{2.108}
$$

$$\forall\ i\in N^L, k\in J_i, t\in N\setminus\{n\}, v\in V, \alpha'\in A_{ik}, \beta'\in B_{ik}, \gamma'\in\Gamma_{ik}\setminus\{0\};$$

$$
\begin{aligned}
\text{(2) backhaul:}\quad & \sum_{j\in N^B}\sum_{q\in J_j}\sum_{\{u\in N|u\le t\}} \\
& \sum_{\left\{\alpha\in A_{jq}\middle|\alpha'-\widehat{l}_{jq}+1\le\alpha\le\alpha'+\widehat{l}_{ik}-1\right\}} \\
& \sum_{\left\{\beta\in B_{jq}\middle|\beta'-\widehat{w}_{jq}+1\le\beta\le\beta'+\widehat{w}_{ik}-1\right\}} f_{jq}\cdot\pi^{jquv}_{\alpha\beta(\gamma'-h_{jq})} \\
& \le \left(1-(1-f_{ik})\cdot\pi^{iktv}_{\alpha'\beta'\gamma'}\right)\cdot M_2
\end{aligned}
\tag{2.109}
$$

$$\forall\ i\in N^B, k\in J_i, t\in N\setminus\{n\}, v\in V, \alpha'\in A_{ik}, \beta'\in B_{ik}, \gamma'\in\Gamma_{ik}\setminus\{0\}.$$

Detailed explanations of the terms on the left- and right-hand side of the formu-

lations are provided in Chapter 2.2.1 for the 3L-VRPTW. They apply here analogously. In (2.108), those items are considered on the left-hand side that are delivered at the same stage as or after the reference item I_{ik} (as in (2.69)), whereas those items are considered that are collected at the same stage as or *before* the reference item in (2.109).

LIFO (P7a), (P7b)

The LIFO constraint for the linehaul items is analogue to the constraint of the 3L-VRPTW, whereas items collected earlier than the reference item must be considered in the case of backhaul customers:

(1) linehaul:

$$\sum_{j\in N^L}\sum_{q\in J_j}\sum_{\{u\in N|t<u<n\}} \sum_{\left\{\alpha\in A_{jq}\middle|\alpha'-\widehat{l}_{jq}+1\le\alpha\le\alpha'+\widehat{l}_{ik}-1\right\}}\sum_{\left\{\beta\in B_{jq}\middle|\beta'-\widehat{w}_{jq}+1\le\beta\le\beta'+\widehat{w}_{ik}-1\right\}}\sum_{\left\{\gamma\in\Gamma_{jq}\middle|\gamma\ge\gamma'+h_{ik}\right\}}\pi_{\alpha\beta\gamma}^{jquv} \le \left(1-\pi_{\alpha'\beta'\gamma'}^{iktv}\right)\cdot M_3 \quad (2.110)$$

$$\forall\ i\in N^L, k\in J_i, t\in N\setminus\{n\}, v\in V, \alpha'\in A_{ik}, \beta'\in B_{ik}, \gamma'\in\Gamma_{ik},$$

$$\sum_{j\in N^L}\sum_{q\in J_j}\sum_{\{u\in N|t<u<n\}} \sum_{\left\{\alpha\in A_{jq}\middle|\alpha\ge\alpha'+\widehat{l}_{ik}\right\}}\sum_{\left\{\beta\in B_{jq}\middle|\beta'-\widehat{w}_{jq}+1\le\beta\le\beta'+\widehat{w}_{ik}-1\right\}}\sum_{\left\{\gamma\in\Gamma_{jq}\middle|\gamma'-h_{jq}+1\le\gamma\le\gamma'+h_{ik}-1\right\}}\pi_{\alpha\beta\gamma}^{jquv} \le \left(1-\pi_{\alpha'\beta'\gamma'}^{iktv}\right)\cdot M_3 \quad (2.111)$$

$$\forall\ i\in N^L, k\in J_i, t\in N\setminus\{n\}, v\in V, \alpha'\in A_{ik}, \beta'\in B_{ik}, \gamma'\in\Gamma_{ik};$$

(2) backhaul:

$$\sum_{j\in N^B}\sum_{q\in J_j}\sum_{\{u\in N|u<t\}} \sum_{\left\{\alpha\in A_{jq}\middle|\alpha'-\widehat{l}_{jq}+1\le\alpha\le\alpha'+\widehat{l}_{ik}-1\right\}}\sum_{\left\{\beta\in B_{jq}\middle|\beta'-\widehat{w}_{jq}+1\le\beta\le\beta'+\widehat{w}_{ik}-1\right\}}\sum_{\left\{\gamma\in\Gamma_{jq}\middle|\gamma\ge\gamma'+h_{ik}\right\}}\pi_{\alpha\beta\gamma}^{jquv} \le \left(1-\pi_{\alpha'\beta'\gamma'}^{iktv}\right)\cdot M_3 \quad (2.112)$$

$$\forall\ i\in N^B, k\in J_i, t\in N\setminus\{n\}, v\in V, \alpha'\in A_{ik}, \beta'\in B_{ik}, \gamma'\in\Gamma_{ik},$$

$$\sum_{j\in N^B}\sum_{q\in J_j}\sum_{\{u\in N|u<t\}} \tag{2.113}$$
$$\sum_{\left\{\alpha\in A_{jq}\middle|\alpha\geq\alpha'+\widehat{l}_{ik}\right\}}\sum_{\left\{\beta\in B_{jq}\middle|\beta'-\widehat{w}_{jq}+1\leq\beta\leq\beta'+\widehat{w}_{ik}-1\right\}}\sum_{\left\{\gamma\in\Gamma_{jq}\middle|\gamma'-h_{jq}+1\leq\gamma\leq\gamma'+h_{ik}-1\right\}}\pi^{jquv}_{\alpha\beta\gamma}$$
$$\leq\left(1-\pi^{iktv}_{\alpha'\beta'\gamma'}\right)\cdot M_3$$
$$\forall\ i\in N^B, k\in J_i, t\in N\setminus\{n\}, v\in V, \alpha'\in A_{ik}, \beta'\in B_{ik}, \gamma'\in\Gamma_{ik}.$$

(2.110) and (2.112) refer to the LIFO constraint along the γ-axis (analogue to (2.70)) and (2.111) and (2.113) refer to the LIFO constraint along the α-axis (analogue to (2.71)).

Reachability (P8)

The reachability constraint (2.72) can be adopted unchanged for both linehaul and backhaul customers. The determination of δ_{ik} needs to be adjusted:

(1) linehaul:

$$\begin{aligned}\delta_{ik}=\sum_{t\in N\setminus\{n\}}\sum_{v\in V}\Bigg(\sum_{\alpha\in A_{ik}}\sum_{\beta\in B_{ik}}\sum_{\gamma\in\Gamma_{ik}}\Bigg(\pi^{iktv}_{\alpha\beta\gamma}\cdot\max\Bigg(\alpha+\widehat{l}_{ik},\\ \max_{\substack{j\in N^L\setminus\{i\},\\ q\in J_j}}\Bigg(\sum_{\{u\in N|t<u<n\}}\sum_{\left\{\alpha\in A_{jq}\middle|\alpha\geq\alpha+\widehat{l}_{ik}-\widehat{l}_{jq}+1\right\}}\\ \sum_{\left\{\beta\in B_{ik}\middle|\beta-\widehat{w}_{jq}+1\leq\beta\leq\beta+\widehat{w}_{ik}-1\right\}}\sum_{\left\{\gamma\in\Gamma_{jq}\middle|\gamma\leq\gamma-h_{jq}\right\}}\pi^{jquv}_{\alpha\beta\gamma}\cdot\left(\alpha+\widehat{l}_{jq}\right)\Bigg),\\ \max_{q\in J_i\setminus\{k\}}\Bigg(\sum_{\left\{\alpha\in A_{iq}\middle|\alpha+\widehat{l}_{ik}-\widehat{l}_{iq}+1\leq\alpha_{iq}\leq\alpha+\widehat{l}_{ik}-1\right\}}\sum_{\left\{\beta\in B_{iq}\middle|\beta-\widehat{w}_{iq}+1\leq\beta_{iq}\leq\beta+\widehat{w}_{ik}-1\right\}}\\ \sum_{\left\{\gamma\in\Gamma_{iq}\middle|\gamma\leq\gamma-h_{iq}\right\}}\pi^{iqtv}_{\alpha\beta\gamma}\cdot\left(\alpha+\widehat{l}_{iq}\right)\Bigg)\Bigg)\Bigg)\Bigg)\Bigg)\Bigg)\end{aligned} \tag{2.114}$$
$$\forall\ i\in N^L, k\in J_i;$$

(2) backhaul:

$$\delta_{ik} = \sum_{t \in N \setminus \{n\}} \sum_{v \in V} \left(\sum_{\alpha \in A_{ik}} \sum_{\beta \in B_{ik}} \sum_{\gamma \in \Gamma_{ik}} \left(\pi_{\alpha\beta\gamma}^{iktv} \cdot \max \left(\alpha + \widehat{l}_{ik}, \right. \right. \right.$$

$$\max_{\substack{j \in N^B \setminus \{i\}, \\ q \in J_j}} \left(\sum_{\{u \in N | u < t\}} \sum_{\left\{ \alpha \in A_{jq} \middle| \alpha \geq \alpha + \widehat{l}_{ik} - \widehat{l}_{jq} + 1 \right\}} \right.$$

$$\left. \sum_{\left\{ \beta \in B_{ik} \middle| \beta - \widehat{w}_{jq} + 1 \leq \beta \leq \beta + \widehat{w}_{ik} - 1 \right\}} \sum_{\left\{ \gamma \in \Gamma_{jq} \middle| \gamma \leq \gamma - h_{jq} \right\}} \pi_{\alpha\beta\gamma}^{jquv} \cdot \left(\alpha + \widehat{l}_{jq} \right) \right), \tag{2.115}$$

$$\max_{q \in J_i \setminus \{k\}} \left(\sum_{\left\{ \alpha \in A_{iq} \middle| \alpha + \widehat{l}_{ik} - \widehat{l}_{iq} + 1 \leq \alpha_{iq} \leq \alpha + \widehat{l}_{ik} - 1 \right\}} \sum_{\left\{ \beta \in B_{iq} \middle| \beta - \widehat{w}_{iq} + 1 \leq \beta_{iq} \leq \beta + \widehat{w}_{ik} - 1 \right\}} \right.$$

$$\left. \left. \left. \left. \sum_{\left\{ \gamma \in \Gamma_{iq} \middle| \gamma \leq \gamma - h_{iq} \right\}} \pi_{\alpha\beta\gamma}^{iqtv} \cdot \left(\alpha + \widehat{l}_{iq} \right) \right) \right) \right) \right)$$

$$\forall\ i \in N^B, k \in J_i.$$

Whereas the determination of δ_{ik} is similar to (2.73) in the case of linehaul customers (only the domain sets for the customer indices are adapted), the equation for backhaul customers is also changed with respect to the domains of the stage indices. Here, items need to be considered that are picked up before the reference item ($u < t$). A detailed explanation of the composition of δ_{ik} is presented in Chapter 2.2.1.

Robust stability (P9)

Similarly to the vertical stability constraint, only minor changes have to be made for the robust stability constraint. In particular, the domains of the customer indices need to be adapted to the customer sets N^L and N^B:

(1) linehaul:

$$\sum_{\left\{ \alpha \in A_0 \middle| \alpha' \leq \alpha \leq \alpha' + \widehat{l}_{ik} - 1 \right\}} \sum_{\left\{ \beta \in B_0 \middle| \beta' \leq \beta \leq \beta' + \widehat{w}_{ik} - 1 \right\}} \sigma_{\alpha\beta\gamma'}^{vL} \geq VSP \cdot l_{ik} \cdot w_{ik} \cdot \pi_{\alpha'\beta'\gamma'}^{iktv} \tag{2.116}$$

$$\forall\, i \in N^L, k \in J_i, t \in N \setminus \{n\}, v \in V, \alpha' \in A_{ik}, \beta' \in B_{ik}, \gamma' \in \Gamma_{ik} \setminus \{0\},$$

where the binary support variable $\sigma^{vL}_{\alpha\beta\gamma}$ exclusively considers linehaul items:

$$\sigma^{vL}_{\alpha\beta\gamma} = \begin{cases} 1, & \text{if } \sum_{i\in N^L} \sum_{k\in J_i} \sum_{\{\gamma'\in\Gamma_{ik}|\gamma'<\gamma\}} \varphi^{ikv}_{\alpha\beta\gamma} = \gamma, \text{ or } \gamma = 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.117}$$

$$\forall\, v \in V, \alpha \in A_0, \beta \in B_0, \gamma \in \Gamma_0;$$

(2) backhaul:

$$\sum_{\left\{\alpha\in A_0 \middle| \alpha'\le\alpha\le\alpha'+\widehat{l}_{ik}-1\right\}} \sum_{\left\{\beta\in B_0 \middle| \beta'\le\beta\le\beta'+\widehat{w}_{ik}-1\right\}} \sigma^{vB}_{\alpha\beta\gamma'} \ge VSP \cdot l_{ik} \cdot w_{ik} \cdot \sum_{t\in N\setminus\{n\}} \pi^{iktv}_{\alpha'\beta'\gamma'} \tag{2.118}$$

$$\forall\, i \in N^B, k \in J_i, v \in V, \alpha' \in A_{ik}, \beta' \in B_{ik}, \gamma' \in \Gamma_{ik} \setminus \{0\},$$

where the binary support variable $\sigma^{vB}_{\alpha\beta\gamma}$ exclusively considers backhaul items:

$$\sigma^{vB}_{\alpha\beta\gamma} = \begin{cases} 1, & \text{if } \sum_{i\in N^B} \sum_{k\in J_i} \sum_{\{\gamma'\in\Gamma_{ik}|\gamma'<\gamma\}} \varphi^{ikv}_{\alpha\beta\gamma} = \gamma, \text{ or } \gamma = 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.119}$$

$$\forall\, v \in V, \alpha \in A_0, \beta \in B_0, \gamma \in \Gamma_0.$$

Load bearing strength (P10)

Applying the load bearing strength constraint (2.78) (based on the assumptions of Ratcliff and Bischoff, 1998) to items of the linehaul customers, only other items of linehaul customers are considered:

(1) linehaul:

$$\begin{aligned} &\sum_{j\in N^L} \sum_{q\in J_j} \sum_{u\in N\setminus\{n\}} \\ &\sum_{\left\{\alpha\in A_{jq} \middle| \alpha'-\widehat{l}_{jq}+1\le\alpha\le\alpha'\right\}} \sum_{\left\{\beta\in B_{jq} \middle| \beta'-\widehat{w}_{jq}+1\le\beta\le\beta'\right\}} \sum_{\left\{\gamma\in\Gamma_{jq} \middle| \gamma'+1\le\gamma\le H-h_{jq}\right\}} \left(\frac{d_{jq}}{l_{jq}\cdot w_{jq}}\right) \cdot \pi^{jquv}_{\alpha\beta\gamma} \\ \le &\sum_{i\in N^L} \sum_{k\in J_i} \sum_{t\in N\setminus\{n\}} \\ &\sum_{\left\{\alpha\in A_{ik} \middle| \alpha'-\widehat{l}_{ik}+1\le\alpha\le\alpha'\right\}} \sum_{\left\{\beta\in B_{ik} \middle| \beta'-\widehat{w}_{ik}+1\le\beta\le\beta'\right\}} \sum_{\left\{\gamma\in\Gamma_{ik} \middle| \gamma'-h_{ik}+1\le\gamma\le\gamma'\right\}} p_{ik} \cdot \pi^{iktv}_{\alpha\beta\gamma} \end{aligned} \tag{2.120}$$

$$\forall\ v \in V, \alpha' \in A_0, \beta' \in B_0, \gamma' \in \Gamma_0;$$

Similarly, only other items of backhaul customers are considered if the constraint is applied to items of backhaul customers:

(2) backhaul:

$$\begin{aligned}
&\sum_{j\in N^B}\sum_{q\in J_j}\sum_{u\in N\setminus\{n\}} \\
&\sum_{\left\{\alpha\in A_{jq}\middle|\alpha'-\widehat{l}_{jq}+1\le\alpha\le\alpha'\right\}}\ \sum_{\left\{\beta\in B_{jq}\middle|\beta'-\widehat{w}_{jq}+1\le\beta\le\beta'\right\}}\ \sum_{\left\{\gamma\in\Gamma_{jq}\middle|\gamma'+1\le\gamma\le H-h_{jq}\right\}} \left(\frac{d_{jq}}{l_{jq}\cdot w_{jq}}\right)\cdot \pi_{\alpha\beta\gamma}^{jquv} \\
\le &\sum_{i\in N^B}\sum_{k\in J_i}\sum_{t\in N\setminus\{n\}} \\
&\sum_{\left\{\alpha\in A_{ik}\middle|\alpha'-\widehat{l}_{ik}+1\le\alpha\le\alpha'\right\}}\ \sum_{\left\{\beta\in B_{ik}\middle|\beta'-\widehat{w}_{ik}+1\le\beta\le\beta'\right\}}\ \sum_{\left\{\gamma\in\Gamma_{ik}\middle|\gamma'-h_{ik}+1\le\gamma\le\gamma'\right\}} p_{ik}\cdot\pi_{\alpha\beta\gamma}^{iktv}
\end{aligned} \tag{2.121}$$

$$\forall\ v \in V, \alpha' \in A_0, \beta' \in B_0, \gamma' \in \Gamma_0.$$

Analogously, the extended variant of the load bearing strength constraint is formulated as follows:

(1) linehaul:

$$\begin{aligned}
&\sum_{j\in N^L}\sum_{q\in J_j}\sum_{u\in N\setminus\{n\}} \sum_{\left\{\alpha\in A_{jq}\middle|\alpha'-\widehat{l}_{jq}+1\le\alpha\le\alpha'\right\}}\ \sum_{\left\{\beta\in B_{jq}\middle|\beta'-\widehat{w}_{jq}+1\le\beta\le\beta'\right\}} \frac{\widehat{d}_{jq}}{a_{jq}}\cdot\pi_{\alpha\beta(\gamma'+h_{ik})}^{jquv} \\
\le &\sum_{\left\{\alpha\in A_{ik}\middle|\alpha'-\widehat{l}_{ik}+1\le\alpha\le\alpha'\right\}}\ \sum_{\left\{\beta\in B_{ik}\middle|\beta'-\widehat{w}_{ik}+1\le\beta\le\beta'\right\}} p_{ik}\cdot\pi_{\alpha\beta\gamma'}^{iktv} \\
&+\left(1-\sum_{\left\{\alpha\in A_{ik}\middle|\alpha'-\widehat{l}_{ik}+1\le\alpha\le\alpha'\right\}}\ \sum_{\left\{\beta\in B_{ik}\middle|\beta'-\widehat{w}_{ik}+1\le\beta\le\beta'\right\}} \pi_{\alpha\beta\gamma'}^{iktv}\right)\cdot M_4
\end{aligned} \tag{2.122}$$

$$\forall\ i \in N^L, k \in J_i, v \in V, t \in N\setminus\{n\}, \alpha' \in A_0, \beta' \in B_0, \gamma' \in \Gamma_0,$$

where

$$\widehat{d}_{ik} = d_{ik} + \sum_{j\in N^L}\sum_{q\in J_j}\left(\frac{supp_{jqik}}{a_{jq}}\cdot\widehat{d}_{jq}\right) \qquad \forall\ i \in N^L, k \in J_i; \tag{2.123}$$

(2) backhaul:

$$\sum_{j\in N^B}\sum_{q\in J_j}\sum_{u\in N\setminus\{n\}}\sum_{\left\{\alpha\in A_{jq}\middle|\alpha'-\widehat{l}_{jq}+1\le\alpha\le\alpha'\right\}}\sum_{\left\{\beta\in B_{jq}\middle|\beta'-\widehat{w}_{jq}+1\le\beta\le\beta'\right\}}\frac{\widehat{d}_{jq}}{a_{jq}}\cdot\pi^{jquv}_{\alpha\beta(\gamma'+h_{ik})}$$

$$\le\sum_{\left\{\alpha\in A_{ik}\middle|\alpha'-\widehat{l}_{ik}+1\le\alpha\le\alpha'\right\}}\sum_{\left\{\beta\in B_{ik}\middle|\beta'-\widehat{w}_{ik}+1\le\beta\le\beta'\right\}}p_{ik}\cdot\pi^{iktv}_{\alpha\beta\gamma'} \tag{2.124}$$

$$+\left(1-\sum_{\left\{\alpha\in A_{ik}\middle|\alpha'-\widehat{l}_{ik}+1\le\alpha\le\alpha'\right\}}\sum_{\left\{\beta\in B_{ik}\middle|\beta'-\widehat{w}_{ik}+1\le\beta\le\beta'\right\}}\pi^{iktv}_{\alpha\beta\gamma'}\right)\cdot M_4$$

$$\forall\ i\in N^B, k\in J_i, v\in V, t\in N\setminus\{n\}, \alpha'\in A_0, \beta'\in B_0, \gamma'\in \Gamma_0,$$

where

$$\widehat{d}_{ik}=d_{ik}+\sum_{j\in N^B}\sum_{q\in J_j}\left(\frac{supp_{jqik}}{a_{jq}}\cdot\widehat{d}_{jq}\right)\qquad\forall\ i\in N^B, k\in J_i. \tag{2.125}$$

$supp_{ikjq}$ $(i,j\in N^L, k\in J_i, q\in J_j)$ can be determined as in (2.63) or (2.64), and a_{ik} $(i\in N^L, k\in J_i)$ as in (2.105). $supp_{ikjq}$ $(i,j\in N^B, k\in J_i, q\in J_j)$ can be determined as in (2.63) or (2.107), and a_{ik} $(i\in N^B, k\in J_i)$ as in (2.106). The formulations for linehaul and backhaul customers differ only in the usage of the customer sets.

Variable domain constraints

The variable domain constraints (2.81)-(2.93) can be adopted unaltered from the model for the 3L-VRPTW. Analogue to $\sigma^v_{\alpha\beta\gamma}$ (see (2.94)), $\sigma^{vL}_{\alpha\beta\gamma}$ and $\sigma^{vB}_{\alpha\beta\gamma}$ are also binary variables:

$$\sigma^{vL}_{\alpha\beta\gamma}\in\{0,1\}\qquad\forall\ v\in V, \alpha\in A_0, \beta\in B_0, \gamma\in\Gamma_0, \tag{2.126}$$

$$\sigma^{vB}_{\alpha\beta\gamma}\in\{0,1\}\qquad\forall\ v\in V, \alpha\in A_0, \beta\in B_0, \gamma\in\Gamma_0. \tag{2.127}$$

2.2.3 The 3L-VRP with mixed backhauls and time windows

While the constraints presented for the 3L-VRPTW can be applied rather easily to the 3L-VRPCB(TW), the formulations for the 3L-VRPMB(TW) (and also for the 3L-VRPSDP(TW)) are more complex. In these problem variants, linehaul and

backhaul items can be in the vehicle at the same time and items can be removed and/or added at every stage of a route. Therefore, linehaul and backhaul items must be considered simultaneously for the formulations of some constraints.

Sets, constants and decision variables

The same sets, constants and decision variables as in the 3L-VRPCB(TW) model are applied.

Model

The objective function contains the minimization of the total routing cost and is adopted from the 3L-VRPTW (2.28).

Routing constraints

The routing constraints (2.29)-(2.34), (2.37), and (2.38), as well as the time window constraints (2.39)-(2.42) from the 3L-VRPTW can be applied equally to the 3L-VRPMB(TW). The weight and volume constraints, however, have to be formulated for every stage of a route, as explained in Chapter 2.1.2.3. The weight constraint can be stated as:

$$\sum_{i \in N^L} \sum_{k \in J_i} \sum_{j \in N_c} \sum_{\{u \in N | u > t\}} d_{ik} \cdot x_{ij}^{uv} + \sum_{i \in N^B} \sum_{k \in J_i} \sum_{j \in N_c} \sum_{\{u \in N | 0 < u \leq t\}} d_{ik} \cdot x_{ij}^{uv} \leq D \qquad (2.128)$$

$$\forall\ v \in V, t \in N.$$

That is, for $t = 0$ the constraint ensures that all linehaul items do not exceed the vehicle weight capacities. For every stage t that follows, (2.128) guarantees that the capacities are observed for all linehaul items that are delivered after stage t and all backhaul items that are collected before and in stage t. In the last stage of the route, all backhaul items must not exceed the vehicle capacity. Analogously, the volume constraint can be formulated as:

$$\begin{aligned} &\sum_{i \in N^L} \sum_{k \in J_i} \sum_{j \in N_c} \sum_{\{u \in N | u > t\}} l_{ik} \cdot w_{ik} \cdot h_{ik} \cdot x_{ij}^{uv} \\ &+ \sum_{i \in N^B} \sum_{k \in J_i} \sum_{j \in N_c} \sum_{\{j \in N | 0 < u \leq t\}} l_{ik} \cdot w_{ik} \cdot h_{ik} \cdot x_{ij}^{uv} \leq L \cdot W \cdot H \end{aligned} \qquad (2.129)$$

$$\forall\ v \in V, t \in N.$$

Packing constraints

Geometrical constraints (P1)-(P3)

Constraint (2.45) – guaranteeing that all items are placed completely inside the loading space – from the 3L-VRPTW can also be applied for the 3L-VRPMB(TW). The non-overlapping constraint has to be satisfied at every stage of a route. It can be formulated as:

$$\begin{aligned}
&\sum_{i\in N^L}\sum_{k\in J_i}\sum_{\{u\in N|t\le u<n\}} \\
&\quad\sum_{\left\{\alpha\in A_{ik}\left|\alpha'-\widehat{l}_{ik}+1\le\alpha\le\alpha'\right.\right\}}\ \sum_{\left\{\beta\in B_{ik}\left|\beta'-\widehat{w}_{ik}+1\le\beta\le\beta'\right.\right\}}\ \sum_{\left\{\gamma\in \Gamma_{ik}\left|\gamma'-h'_{ik}+1\le\gamma\le\gamma'\right.\right\}} \pi^{ikuv}_{\alpha\beta\gamma} \\
&+\sum_{i\in N^B}\sum_{k\in J_i}\sum_{\{u\in N|u<t\}} \\
&\quad\sum_{\left\{\alpha\in A_{ik}\left|\alpha'-\widehat{l}_{ik}+1\le\alpha\le\alpha'\right.\right\}}\ \sum_{\left\{\beta\in B_{ik}\left|\beta'-\widehat{w}_{ik}+1\le\beta\le\beta'\right.\right\}}\ \sum_{\left\{\gamma\in \Gamma_{ik}\left|\gamma'-h'_{ik}+1\le\gamma\le\gamma'\right.\right\}} \pi^{ikuv}_{\alpha\beta\gamma} \le 1
\end{aligned} \tag{2.130}$$

$$\forall\ v\in V, t\in N, \alpha'\in A_0, \beta'\in B_0, \gamma'\in \Gamma_0.$$

For a certain stage t, those linehaul items must be considered, that are delivered in stage t or later, and those backhaul items, that are collected earlier than in stage t.

LIFO (P7a), (P7b), (P7c)

The formulations of the LIFO constraint presented for the 3L-VRPCB(TW) are also used for the 3L-VRPMB(TW). This includes (2.110) and (2.111), which consider *only* linehaul items, and (2.112) and (2.113), which consider *only* backhaul items. However, the LIFO constraint must also regard linehaul and backhaul items that are in the vehicle at the same time.

Firstly, linehaul and backhaul items must not be placed above each other, i.e. a linehaul item I_{ik} delivered in stage t must not be placed under a backhaul item I_{jq} that is collected before I_{ik} is delivered (in stage $u < t$):

$$\sum_{j\in N^B}\sum_{q\in J_j}\sum_{\{u\in N|u<t\}}$$
$$\sum_{\left\{\alpha\in A_{jq}\middle|\alpha'-\widehat{l}_{jq}+1\leq\alpha\leq\alpha'+\widehat{l}_{ik}-1\right\}}\ \sum_{\left\{\beta\in B_{jq}\middle|\beta'-\widehat{w}_{jq}+1\leq\beta\leq\beta'+\widehat{w}_{ik}-1\right\}}\ \sum_{\left\{\gamma\in\Gamma_{jq}\middle|\gamma\geq\gamma'+h'_{ik}\right\}}\pi^{jquv}_{\alpha\beta\gamma}$$
$$\leq\left(1-\pi^{iktv}_{\alpha'\beta'\gamma'}\right)\cdot M_3 \tag{2.131}$$

$$\forall\ i\in N^L, k\in J_i, v\in V, t\in N\setminus\{n\}, \alpha'\in A_{ik}, \beta'\in B_{ik}, \gamma\in\Gamma_{ik}.$$

Vice versa, a backhaul item I_{ik} must not be placed under a linehaul item I_{jq} that is delivered after I_{ik} is collected:

$$\sum_{j\in N^L}\sum_{q\in J_j}\sum_{\{u\in N|t<u<n\}}$$
$$\sum_{\left\{\alpha\in A_{jq}\middle|\alpha'-\widehat{l}_{jq}+1\leq\alpha\leq\alpha'+\widehat{l}_{ik}-1\right\}}\ \sum_{\left\{\beta\in B_{jq}\middle|\beta'-\widehat{w}_{jq}+1\leq\beta\leq\beta'+\widehat{w}_{ik}-1\right\}}\ \sum_{\left\{\gamma\in\Gamma_{jq}\middle|\gamma\geq\gamma'+h'_{ik}\right\}}\pi^{jquv}_{\alpha\beta\gamma}$$
$$\leq\left(1-\pi^{iktv}_{\alpha'\beta'\gamma'}\right)\cdot M_3 \tag{2.132}$$

$$\forall\ i\in N^B, k\in J_i, v\in V, t\in N\setminus\{n\}, \alpha'\in A_{ik}, \beta'\in B_{ik}, \gamma\in\Gamma_{ik}.$$

For the placement along the α-axis it can be stated analogously that a linehaul item I_{ik} must not be placed behind a backhaul item I_{jq} that is collected before I_{ik} is delivered:

$$\sum_{j\in N^B}\sum_{q\in J_j}\sum_{\{u\in N|u<t\}}$$
$$\sum_{\left\{\alpha\in A_{jq}\middle|\alpha\geq\alpha'+\widehat{l}_{ik}\right\}}\ \sum_{\left\{\beta\in B_{jq}\middle|\beta'-\widehat{w}_{jq}+1\leq\beta\leq\beta'+\widehat{w}_{ik}-1\right\}}\ \sum_{\left\{\gamma\in\Gamma_{jq}\middle|\gamma'-h_{jq}+1\leq\gamma\leq\gamma'+h'_{ik}-1\right\}}\pi^{jquv}_{\alpha\beta\gamma}$$
$$\leq\left(1-\pi^{iktv}_{\alpha'\beta'\gamma'}\right)\cdot M_3 \tag{2.133}$$

$$\forall\ i\in N^L, k\in J_i, v\in V, t\in N\setminus\{n\}, \alpha'\in A_{ik}, \beta'\in B_{ik}, \gamma\in\Gamma_{ik},$$

and that a backhaul item I_{ik} must not be placed behind a linehaul item I_{jq} that is delivered after I_{ik} is collected:

$$\sum_{j \in N^L} \sum_{q \in J_j} \sum_{\{u \in N | t<u<n\}} \sum_{\left\{\alpha \in A_{jq} \middle| \alpha \geq \alpha' + \widehat{l}_{ik}\right\}} \sum_{\left\{\beta \in B_{jq} \middle| \beta' - \widehat{w}_{jq} + 1 \leq \beta \leq \beta' + \widehat{w}_{ik} - 1\right\}} \sum_{\left\{\gamma \in \Gamma_{jq} \middle| \gamma' - h_{jq} + 1 \leq \gamma \leq \gamma' + h'_{ik} - 1\right\}} \pi^{jquv}_{\alpha\beta\gamma}$$
$$\leq \left(1 - \pi^{iktv}_{\alpha'\beta'\gamma'}\right) \cdot M_3 \tag{2.134}$$

$$\forall\, i \in N^B, k \in J_i, v \in V, t \in N \setminus \{n\}, \alpha' \in A_{ik}, \beta' \in B_{ik}, \gamma \in \Gamma_{ik}.$$

Other packing constraints

Since linehaul and backhaul items, that are in the vehicle at the same time, cannot be placed above or in front of each other, the formulations from the 3L-VRPCB(TW) for the vertical stability constraint (2.104)-(2.107), the fragility constraint (2.108) and (2.109), the reachability constraint (2.72) with (2.114) and (2.115), the robust stability constraint (2.116)-(2.119), and the load bearing strength constraint (2.120)-(2.125) are applied to the 3L-VRPMB(TW), too.

Variable domain constraints

The variable domain constraints (2.81)-(2.93) and (2.126)-(2.127) can be adopted unaltered from the models for the 3L-VRPTW and 3L-VRPCB(TW).

2.2.4 The 3L-VRP with simultaneous delivery and pickup and time windows

Sets, constants and decision variables

Mostly the same constants and sets as before are applied (see Table 2.3). Additionally, m_i^L and m_i^B denote the numbers of linehaul items demanded by customer i and the numbers of backhaul items supplied by customer i. The respective items are contained in the sets J_i^L and J_i^B. The decision variables introduced above are used.

Model

The previously introduced objective function (2.28) is applied. The further model for the 3L-VRPSDP(TW) is similar to the model for the 3L-VRPMB(TW). The difference is that the sets of items are divided into sets of linehaul items and sets of backhaul items instead of dividing the customers.

Routing constraints

The routing constraints (2.29)-(2.34), (2.37), and (2.38), and the time window constraints (2.39)-(2.42) are applied unchanged to the 3L-VRPSDP(TW).

The weight constraint is adapted:

$$\sum_{i \in N_c} \sum_{k \in J_i^L} \sum_{j \in N} \sum_{\{u \in N | u > t\}} d_{ik} \cdot x_{ij}^{uv} + \sum_{i \in N_c} \sum_{k \in J_i^B} \sum_{j \in N} \sum_{\{u \in N | 0 < u \leq t\}} d_{ik} \cdot x_{ij}^{uv} \leq D \quad \forall\ v \in V, t \in N. \tag{2.135}$$

That is, at every stage t the total weight of all linehaul items delivered after stage t and all backhaul items collected at stage t and earlier are considered. Likewise, the volume constraint can be stated as:

$$\sum_{i \in N_c} \sum_{k \in J_i^L} \sum_{j \in N} \sum_{\{u \in N | u > t\}} l_{ik} \cdot w_{ik} \cdot h_{ik} \cdot x_{ij}^{uv} + \sum_{i \in N_c} \sum_{k \in J_i^B} \sum_{j \in N} \sum_{\{u \in N | 0 < u \leq t\}} l_{ik} \cdot w_{ik} \cdot h_{ik} \cdot x_{ij}^{uv} \leq L \cdot W \cdot H \quad \forall\ v \in V, t \in N. \tag{2.136}$$

Packing constraints

The packing constraints formulated for the 3L-VRPSDP(TW) are based on the formulations proposed for the 3L-VRPCB(TW) and the 3L-VRPMB(TW). As in case of the 3L-VRPMB(TW), the constraints regarding the stacking of items (fragility, stability, load bearing) and the reachability constraint can still be formulated separately for linehaul and backhaul items as the different types cannot be placed above or in front of each other. Merely, the customer and item sets need to be adjusted.

Geometrical constraints (P1)-(P3)

The previously formulated constraint regarding the complete placement inside the loading space (2.45) can be used equally for the 3L-VRPSDP(TW). The non-overlapping constraint can be formulated similarly to the respective constraint for the 3L-VRPMB(TW) (2.130) by exchanging the domains for the customer (i) and the

item (k) indices:

$$\sum_{i\in N_c}\sum_{k\in J_i^L}\sum_{\{u\in N|t\le u<n\}} \sum_{\left\{\alpha\in A_{ik}\left|\alpha'-\widehat{l}_{ik}+1\le\alpha\le\alpha'\right.\right\}}\sum_{\left\{\beta\in B_{ik}\left|\beta'-\widehat{w}_{ik}+1\le\beta\le\beta'\right.\right\}}\sum_{\left\{\gamma\in\Gamma_{ik}\left|\gamma'-h'_{ik}+1\le\gamma\le\gamma'\right.\right\}}\pi_{\alpha\beta\gamma}^{ikuv} + \sum_{i\in N_c}\sum_{k\in J_i^B}\sum_{\{u\in N|u<t\}} \sum_{\left\{\alpha\in A_{ik}\left|\alpha'-\widehat{l}_{ik}+1\le\alpha\le\alpha'\right.\right\}}\sum_{\left\{\beta\in B_{ik}\left|\beta'-\widehat{w}_{ik}+1\le\beta\le\beta'\right.\right\}}\sum_{\left\{\gamma\in\Gamma_{ik}\left|\gamma'-h'_{ik}+1\le\gamma\le\gamma'\right.\right\}}\pi_{\alpha\beta\gamma}^{ikuv} \le 1 \tag{2.137}$$

$$\forall\ \alpha'\in A_0,\beta'\in B_0,\gamma'\in\Gamma_0,v\in V,t\in N.$$

Vertical stability (P5)

Similarly to (2.104)-(2.106), the vertical stability constraint can be formulated as:

$$a_{ik}\ge VSP\cdot l_{ik}\cdot w_{ik}\qquad\forall\ i\in N_c,k\in I_i. \tag{2.138}$$

The determination of the supported area of an item is adjusted to the item sets:

(1) linehaul:

$$a_{ik}=l_{ik}\cdot w_{ik}\cdot\sum_{\alpha\in A_{ik}}\sum_{\beta\in B_{ik}}\sum_{t\in N\setminus\{n\}}\sum_{v\in V}\pi_{\alpha\beta0}^{iktv}+\sum_{j\in N_c}\sum_{q\in J_j^L}supp_{ikjq} \tag{2.139}$$

$$\forall\ i\in N_c,k\in J_i^L;$$

(2) backhaul:

$$a_{ik}=l_{ik}\cdot w_{ik}\cdot\sum_{\alpha\in A_{ik}}\sum_{\beta\in B_{ik}}\sum_{t\in N\setminus\{n\}}\sum_{v\in V}\pi_{\alpha\beta0}^{iktv}+\sum_{j\in N_c}\sum_{q\in J_j^B}supp_{ikjq} \tag{2.140}$$

$$\forall\ i\in N_c,k\in J_i^B.$$

For the linehaul items, $supp_{ikjq}$ $(i,j\in N_c,k\in J_i^L,q\in J_j^L)$ can be determined analogously to (2.63) or (2.64). For the backhaul items, $supp_{ikjq}$ $(i,j\in N_c,k\in J_i^B,q\in J_j^B)$ can be determined analogously to (2.63) or (2.107).

Fragility (P6)

The fragility constraint can be stated as:

$$\begin{aligned}\text{(1) linehaul:}\quad & \sum_{j\in N_c}\sum_{q\in J_j^L}\sum_{\{u\in N|t\leq u<n\}} \\ & \sum_{\left\{\alpha\in A_{jq}\middle|\alpha'-\widehat{l}_{jq}+1\leq\alpha\leq\alpha'+\widehat{l}_{ik}-1\right\}} \\ & \sum_{\left\{\beta\in B_{jq}\middle|\beta'-\widehat{w}_{jq}+1\leq\beta\leq\beta'+\widehat{w}_{ik}-1\right\}} f_{jq}\cdot\pi^{jquv}_{\alpha\beta(\gamma'-h_{jq})} \\ & \leq\left(1-(1-f_{ik})\cdot\pi^{iktv}_{\alpha'\beta'\gamma'}\right)\cdot M_2 \end{aligned} \tag{2.141}$$

$$\forall\ i\in N_c, k\in J_i^L, t\in N\setminus\{n\}, v\in V, \alpha'\in A_{ik}, \beta'\in B_{ik}, \gamma'\in \Gamma_{ik}\setminus\{0\};$$

$$\begin{aligned}\text{(2) backhaul:}\quad & \sum_{j\in N_c}\sum_{q\in J_j^B}\sum_{\{u\in N|u\leq t\}} \\ & \sum_{\left\{\alpha\in A_{jq}\middle|\alpha'-\widehat{l}_{jq}+1\leq\alpha\leq\alpha'+\widehat{l}_{ik}-1\right\}} \\ & \sum_{\left\{\beta\in B_{jq}\middle|\beta'-\widehat{w}_{jq}+1\leq\beta\leq\beta'+\widehat{w}_{ik}-1\right\}} f_{jq}\cdot\pi^{jquv}_{\alpha\beta(\gamma'-h_{jq})} \\ & \leq\left(1-(1-f_{ik})\cdot\pi^{iktv}_{\alpha'\beta'\gamma'}\right)\cdot M_2 \end{aligned} \tag{2.142}$$

$$\forall\ i\in N_c, k\in J_i^B, t\in N\setminus\{n\}, v\in V, \alpha'\in A_{ik}, \beta'\in B_{ik}, \gamma'\in \Gamma_{ik}\setminus\{0\}.$$

LIFO (P7d)

First of all, the formulations of the LIFO constraint (2.110)-(2.113) from the 3L-VRPCB(TW) and 3L-VRPMB(TW) can be applied and adjusted. They ensure that a linehaul item is not placed above or in front of another linehaul item that is delivered earlier, and that a backhaul item is not placed above or in front of another backhaul item that is collected later:

(1) linehaul:

$$\sum_{j\in N_c}\sum_{q\in J_j^L}\sum_{\{u\in N|t<u<n\}} \sum_{\left\{\alpha\in A_{jq}\middle|\alpha'-\widehat{l}_{jq}+1\le\alpha\le\alpha'+\widehat{l}_{ik}-1\right\}}\sum_{\left\{\beta\in B_{jq}\middle|\beta'-\widehat{w}_{jq}+1\le\beta\le\beta'+\widehat{w}_{ik}-1\right\}}\sum_{\left\{\gamma\in\Gamma_{jq}\middle|\gamma\ge\gamma'+h_{ik}\right\}}\pi_{\alpha\beta\gamma}^{jquv} \le\left(1-\pi_{\alpha'\beta'\gamma'}^{iktv}\right)\cdot M_3 \tag{2.143}$$

$$\forall\ i\in N_c, k\in J_i^L, t\in N\setminus\{n\}, v\in V, \alpha'\in A_{ik}, \beta'\in B_{ik}, \gamma'\in\Gamma_{ik},$$

$$\sum_{j\in N_c}\sum_{q\in J_j^L}\sum_{\{u\in N|t<u<n\}} \sum_{\left\{\alpha\in A_{jq}\middle|\alpha\ge\alpha'+\widehat{l}_{ik}\right\}}\sum_{\left\{\beta\in B_{jq}\middle|\beta'-\widehat{w}_{jq}+1\le\beta\le\beta'+\widehat{w}_{ik}-1\right\}}\sum_{\left\{\gamma\in\Gamma_{jq}\middle|\gamma'-h_{jq}+1\le\gamma\le\gamma'+h_{ik}-1\right\}}\pi_{\alpha\beta\gamma}^{jquv} \le\left(1-\pi_{\alpha'\beta'\gamma'}^{iktv}\right)\cdot M_3 \tag{2.144}$$

$$\forall\ i\in N_c, k\in J_i^L, t\in N\setminus\{n\}, v\in V, \alpha'\in A_{ik}, \beta'\in B_{ik}, \gamma'\in\Gamma_{ik};$$

(2) backhaul:

$$\sum_{j\in N_c}\sum_{q\in J_j^B}\sum_{\{u\in N|u<t\}} \sum_{\left\{\alpha\in A_{jq}\middle|\alpha'-\widehat{l}_{jq}+1\le\alpha\le\alpha'+\widehat{l}_{ik}-1\right\}}\sum_{\left\{\beta\in B_{jq}\middle|\beta'-\widehat{w}_{jq}+1\le\beta\le\beta'+\widehat{w}_{ik}-1\right\}}\sum_{\left\{\gamma\in\Gamma_{jq}\middle|\gamma\ge\gamma'+h_{ik}\right\}}\pi_{\alpha\beta\gamma}^{jquv} \le\left(1-\pi_{\alpha'\beta'\gamma'}^{iktv}\right)\cdot M_3 \tag{2.145}$$

$$\forall\ i\in N_c, k\in J_i^B, t\in N\setminus\{n\}, v\in V, \alpha'\in A_{ik}, \beta'\in B_{ik}, \gamma'\in\Gamma_{ik},$$

$$\sum_{j\in N_c}\sum_{q\in J_j^B}\sum_{\{u\in N|u<t\}} \sum_{\left\{\alpha\in A_{jq}\middle|\alpha\ge\alpha'+\widehat{l}_{ik}\right\}}\sum_{\left\{\beta\in B_{jq}\middle|\beta'-\widehat{w}_{jq}+1\le\beta\le\beta'+\widehat{w}_{ik}-1\right\}}\sum_{\left\{\gamma\in\Gamma_{jq}\middle|\gamma'-h_{jq}+1\le\gamma\le\gamma'+h_{ik}-1\right\}}\pi_{\alpha\beta\gamma}^{jquv} \le\left(1-\pi_{\alpha'\beta'\gamma'}^{iktv}\right)\cdot M_3 \tag{2.146}$$

$$\forall\ i \in N_c, k \in J_i^B, t \in N \setminus \{n\}, v \in V, \alpha' \in A_{ik}, \beta' \in B_{ik}, \gamma' \in \Gamma_{ik}.$$

In order to prevent linehaul and backhaul items from being placed above or in front of each other, the LIFO constraint is formulated similarly to (2.131)-(2.134) from the 3L-VRPMB(TW) model.

(2.147) ensures that no linehaul item is placed under a backhaul item that is collected before the linehaul item is delivered:

$$\begin{aligned}
&\sum_{j \in N_c} \sum_{q \in J_j^B} \sum_{\{u \in N | u < t\}} \\
&\sum_{\left\{\alpha \in A_{jq} \middle| \alpha' - \widehat{l}_{jq} + 1 \leq \alpha \leq \alpha' + \widehat{l}_{ik} - 1\right\}} \sum_{\left\{\beta \in B_{jq} \middle| \beta' - \widehat{w}_{jq} + 1 \leq \beta \leq \beta' + \widehat{w}_{ik} - 1\right\}} \sum_{\left\{\gamma \in \Gamma_{jq} \middle| \gamma \geq \gamma' + h'_{ik}\right\}} \pi_{\alpha\beta\gamma}^{jquv} \\
&\leq \left(1 - \pi_{\alpha'\beta'\gamma'}^{iktv}\right) \cdot M_3
\end{aligned} \tag{2.147}$$

$$\forall\ i \in N_c, k \in J_i^L, v \in V, t \in N \setminus \{n\}, \alpha' \in A_{ik}, \beta' \in B_{ik}, \gamma \in \Gamma_{ik}.$$

(2.148) ensures that no backhaul item is placed under a linehaul item that is delivered after the backhaul item is collected:

$$\begin{aligned}
&\sum_{j \in N_c} \sum_{q \in J_j^L} \sum_{\{u \in N | t < u < n\}} \\
&\sum_{\left\{\alpha \in A_{jq} \middle| \alpha' - \widehat{l}_{jq} + 1 \leq \alpha \leq \alpha' + \widehat{l}_{ik} - 1\right\}} \sum_{\left\{\beta \in B_{jq} \middle| \beta' - \widehat{w}_{jq} + 1 \leq \beta \leq \beta' + \widehat{w}_{ik} - 1\right\}} \sum_{\left\{\gamma \in \Gamma_{jq} \middle| \gamma \geq \gamma' + h'_{ik}\right\}} \pi_{\alpha\beta\gamma}^{jquv} \\
&\leq \left(1 - \pi_{\alpha'\beta'\gamma'}^{iktv}\right) \cdot M_3
\end{aligned} \tag{2.148}$$

$$\forall\ i \in N_c, k \in J_i^B, v \in V, t \in N \setminus \{n\}, \alpha' \in A_{ik}, \beta' \in B_{ik}, \gamma \in \Gamma_{ik}.$$

(2.149) ensures that no linehaul item is placed behind a backhaul item that is collected before the linehaul item is delivered:

$$\begin{aligned}
&\sum_{j \in N_c} \sum_{q \in J_j^B} \sum_{\{u \in N | u < t\}} \\
&\sum_{\left\{\alpha \in A_{jq} \middle| \alpha \geq \alpha' + \widehat{l}_{ik}\right\}} \sum_{\left\{\beta \in B_{jq} \middle| \beta' - \widehat{w}_{jq} + 1 \leq \beta \leq \beta' + \widehat{w}_{ik} - 1\right\}} \sum_{\left\{\gamma \in \Gamma_{jq} \middle| \gamma' - h_{jq} + 1 \leq \gamma \leq \gamma' + h'_{ik} - 1\right\}} \pi_{\alpha\beta\gamma}^{jquv} \\
&\leq \left(1 - \pi_{\alpha'\beta'\gamma'}^{iktv}\right) \cdot M_3
\end{aligned} \tag{2.149}$$

$$\forall\ i \in N_c, k \in J_i^L, v \in V, t \in N \setminus \{n\}, \alpha' \in A_{ik}, \beta' \in B_{ik}, \gamma \in \Gamma_{ik}.$$

(2.148) ensures that no backhaul item is placed behind a linehaul item that is delivered after the backhaul item is collected:

$$\begin{aligned}
&\sum_{j \in N_c} \sum_{q \in J_j^L} \sum_{\{u \in N | t < u < n\}} \\
&\sum_{\left\{\alpha \in A_{jq} \middle| \alpha \geq \alpha' + \widehat{l}_{ik}\right\}} \sum_{\left\{\beta \in B_{jq} \middle| \beta' - \widehat{w}_{jq} + 1 \leq \beta \leq \beta' + \widehat{w}_{ik} - 1\right\}} \sum_{\left\{\gamma \in \Gamma_{jq} \middle| \gamma' - h_{jq} + 1 \leq \gamma \leq \gamma' + h'_{ik} - 1\right\}} \pi_{\alpha\beta\gamma}^{jquv} \\
&\leq \left(1 - \pi_{\alpha'\beta'\gamma'}^{iktv}\right) \cdot M_3
\end{aligned} \tag{2.150}$$

$$\forall\ i \in N_c, k \in J_i^B, v \in V, t \in N \setminus \{n\}, \alpha' \in A_{ik}, \beta' \in B_{ik}, \gamma \in \Gamma_{ik}.$$

Reachability (P8)

The reachability constraint (2.72) is applied to the 3L-VRPSDP(TW). In addition, the positions δ_{ik} $(i \in N_c, k \in J_i)$ for linehaul and backhaul items are determined similarly to those introduced for the 3L-VRPCB(TW) (Chapter 2.2.2):

(1) linehaul:

$$\begin{aligned}
\delta_{ik} = \sum_{t \in N \setminus \{n\}} \sum_{v \in V} \Bigg(& \sum_{\alpha \in A_{ik}} \sum_{\beta \in B_{ik}} \sum_{\gamma \in \Gamma_{ik}} \Bigg(\pi_{\alpha\beta\gamma}^{iktv} \cdot \max \Bigg(\alpha + \widehat{l}_{ik}, \\
& \max_{\substack{j \in N_c \setminus \{i\}, \\ q \in J_j^L}} \Bigg(\sum_{\{u \in N | t < u < n\}} \sum_{\left\{\alpha \in A_{jq} \middle| \alpha \geq \alpha - \widehat{l}_{jq} + 1\right\}} \\
& \sum_{\left\{\beta \in B_{ik} \middle| \beta - \widehat{w}_{jq} + 1 \leq \beta \leq \beta + \widehat{w}_{ik} - 1\right\}} \sum_{\left\{\gamma \in \Gamma_{jq} \middle| \gamma \leq \gamma - h_{jq}\right\}} \pi_{\alpha\beta\gamma}^{jquv} \cdot \left(\alpha + \widehat{l}_{jq}\right) \Bigg), \\
& \max_{q \in J_i^L \setminus \{k\}} \Bigg(\sum_{\left\{\alpha \in A_{iq} \middle| \alpha - \widehat{l}_{iq} + 1 \leq \alpha_{iq} \leq \alpha + \widehat{l}_{ik} - 1\right\}} \sum_{\left\{\beta \in B_{iq} \middle| \beta - \widehat{w}_{iq} + 1 \leq \beta_{iq} \leq \beta + \widehat{w}_{ik} - 1\right\}} \\
& \sum_{\left\{\gamma \in \Gamma_{iq} \middle| \gamma \leq \gamma - h_{iq}\right\}} \pi_{\alpha\beta\gamma}^{iqtv} \cdot \left(\alpha + \widehat{l}_{iq}\right) \Bigg) \Bigg) \Bigg) \Bigg) \Bigg)
\end{aligned} \tag{2.151}$$

$$\forall\ i \in N_c, k \in J_i^L;$$

(2) backhaul:

$$\delta_{ik} = \sum_{t \in N \setminus \{n\}} \sum_{v \in V} \left(\sum_{\alpha \in A_{ik}} \sum_{\beta \in B_{ik}} \sum_{\gamma \in \Gamma_{ik}} \left(\pi_{\alpha\beta\gamma}^{iktv} \cdot \max \left(\alpha + \widehat{l}_{ik}, \right. \right. \right.$$

$$\max_{\substack{j \in N_c \setminus \{i\}, \\ q \in J_j^B}} \left(\sum_{\{u \in N | u < t\}} \sum_{\left\{\alpha \in A_{jq} \middle| \alpha \geq \alpha - \widehat{l}_{jq} + 1\right\}} \right.$$

$$\left. \left. \sum_{\left\{\beta \in B_{ik} \middle| \beta - \widehat{w}_{jq} + 1 \leq \beta \leq \beta + \widehat{w}_{ik} - 1\right\}} \sum_{\left\{\gamma \in \Gamma_{jq} \middle| \gamma \leq \gamma - h_{jq}\right\}} \pi_{\alpha\beta\gamma}^{jquv} \cdot \left(\alpha + \widehat{l}_{jq}\right) \right), \tag{2.152}$$

$$\max_{q \in J_i^B \setminus \{k\}} \left(\sum_{\left\{\alpha \in A_{iq} \middle| \alpha - \widehat{l}_{iq} + 1 \leq \alpha_{iq} \leq \alpha + \widehat{l}_{ik} - 1\right\}} \sum_{\left\{\beta \in B_{iq} \middle| \beta - \widehat{w}_{iq} + 1 \leq \beta_{iq} \leq \beta + \widehat{w}_{ik} - 1\right\}} \right.$$

$$\left. \left. \left. \left. \sum_{\left\{\gamma \in \Gamma_{iq} \middle| \gamma \leq \gamma - h_{iq}\right\}} \pi_{\alpha\beta\gamma}^{iqtv} \cdot \left(\alpha + \widehat{l}_{iq}\right) \right) \right) \right) \right) \right)$$

$$\forall\ i \in N_c, k \in J_i^B.$$

A detailed explanation of the composition of δ_{ik} is presented in connection with the model for the 3L-VRPTW (Chapter 2.2.1).

Robust stability (P9)

The robust stability constraint is as follows:

(1) linehaul:

$$\sum_{\left\{\alpha \in A_0 \middle| \alpha' \leq \alpha \leq \alpha' + \widehat{l}_{ik} - 1\right\}} \sum_{\left\{\beta \in B_0 \middle| \beta' \leq \beta \leq \beta' + \widehat{w}_{ik} - 1\right\}} \sigma_{\alpha\beta\gamma'}^{vL} \geq VSP \cdot l_{ik} \cdot w_{ik} \cdot \sum_{t \in N \setminus \{n\}} \pi_{\alpha'\beta'\gamma'}^{iktv} \tag{2.153}$$

$$\forall\ i \in N_c, k \in J_i^L, v \in V, \alpha' \in A_{ik}, \beta' \in B_{ik}, \gamma' \in \Gamma_{ik} \setminus \{0\},$$

where

$$\sigma^{vL}_{\alpha\beta\gamma} = \max\left(\sum_{i\in N_c}\sum_{k\in J_i^L}\sum_{\{\gamma'\in\Gamma_{ik}|\gamma'<\gamma\}}\varphi^{ikv}_{\alpha\beta\gamma} - \gamma + 1, 0\right) \tag{2.154}$$

$$\forall\ v \in V, \alpha \in A_0, \beta \in B_0, \gamma \in \Gamma_0;$$

(2) backhaul:

$$\sum_{\left\{\alpha\in A_0\middle|\alpha'\le\alpha\le\alpha'+\widehat{l}_{ik}-1\right\}}\ \sum_{\left\{\beta\in B_0\middle|\beta'\le\beta\le\beta'+\widehat{w}_{ik}-1\right\}} \sigma^{vB}_{\alpha\beta\gamma'} \ge VSP\cdot l_{ik}\cdot w_{ik}\cdot \sum_{t\in N\setminus\{n\}}\pi^{iktv}_{\alpha'\beta'\gamma'} \tag{2.155}$$

$$\forall\ i \in N_c, k \in J_i^B, v \in V, \alpha' \in A_{ik}, \beta' \in B_{ik}, \gamma' \in \Gamma_{ik}\setminus\{0\},$$

where

$$\sigma^{vB}_{\alpha\beta\gamma} = \max\left(\sum_{i\in N_c}\sum_{k\in J_i^B}\sum_{\{\gamma'\in\Gamma_{ik}|\gamma'<\gamma\}}\varphi^{ikv}_{\alpha\beta\gamma} - \gamma + 1, 0\right) \tag{2.156}$$

$$\forall\ v \in V, \alpha \in A_0, \beta \in B_0, \gamma \in \Gamma_0.$$

Load bearing strength (P10)

The load bearing strength constraint based on Ratcliff and Bischoff (1998) needs to be formulated for linehaul and backhaul items separately (similarly to the constraint for the 3L-VRPMB(TW)):

(1) linehaul:

$$\begin{aligned}
&\sum_{j\in N_c}\sum_{q\in J_j^L}\sum_{u\in N\setminus\{n\}}\\
&\quad\sum_{\left\{\alpha\in A_{jq}\middle|\alpha'-\widehat{l}_{jq}+1\le\alpha\le\alpha'\right\}}\ \sum_{\left\{\beta\in B_{jq}\middle|\beta'-\widehat{w}_{jq}+1\le\beta\le\beta'\right\}}\ \sum_{\left\{\gamma\in\Gamma_{jq}\middle|\gamma'+1\le\gamma\le H-h_{jq}\right\}}\left(\frac{d_{jq}}{l_{jq}\cdot w_{jq}}\right)\cdot\pi^{jquv}_{\alpha\beta\gamma}\\
\le&\sum_{i\in N_c}\sum_{k\in J_i^L}\sum_{t\in N\setminus\{n\}}\\
&\quad\sum_{\left\{\alpha\in A_{ik}\middle|\alpha'-\widehat{l}_{ik}+1\le\alpha\le\alpha'\right\}}\ \sum_{\left\{\beta\in B_{ik}\middle|\beta'-\widehat{w}_{ik}+1\le\beta\le\beta'\right\}}\ \sum_{\left\{\gamma\in\Gamma_{ik}\middle|\gamma'-h_{ik}+1\le\gamma\le\gamma'\right\}} p_{ik}\cdot\pi^{iktv}_{\alpha\beta\gamma}
\end{aligned} \tag{2.157}$$

$$\forall\ v \in V, \alpha' \in A_0, \beta' \in B_0, \gamma' \in \Gamma_0;$$

(2) backhaul:

$$\begin{aligned}
&\sum_{j\in N_c}\sum_{q\in J_j^B}\sum_{u\in N\setminus\{n\}} \\
&\quad\sum_{\left\{\alpha\in A_{jq}\middle|\alpha'-\widehat{l}_{jq}+1\le\alpha\le\alpha'\right\}}\ \sum_{\left\{\beta\in B_{jq}\middle|\beta'-\widehat{w}_{jq}+1\le\beta\le\beta'\right\}}\ \sum_{\left\{\gamma\in \Gamma_{jq}\middle|\gamma'+1\le\gamma\le H-h_{jq}\right\}} \left(\frac{d_{jq}}{l_{jq}\cdot w_{jq}}\right)\cdot \pi_{\alpha\beta\gamma}^{jquv} \\
\le&\sum_{i\in N_c}\sum_{k\in J_i^B}\sum_{t\in N\setminus\{n\}} \\
&\quad\sum_{\left\{\alpha\in A_{ik}\middle|\alpha'-\widehat{l}_{ik}+1\le\alpha\le\alpha'\right\}}\ \sum_{\left\{\beta\in B_{ik}\middle|\beta'-\widehat{w}_{ik}+1\le\beta\le\beta'\right\}}\ \sum_{\left\{\gamma\in \Gamma_{ik}\middle|\gamma'-h_{ik}+1\le\gamma\le\gamma'\right\}} p_{ik}\cdot\pi_{\alpha\beta\gamma}^{iktv} \qquad (2.158)
\end{aligned}$$

$$\forall\ v\in V, \alpha'\in A_0, \beta'\in B_0, \gamma'\in \Gamma_0.$$

The following formulations consider the alternative, extended load bearing strength constraint:

(1) linehaul:

$$\begin{aligned}
&\sum_{j\in N_c}\sum_{q\in J_j^L}\sum_{u\in N\setminus\{n\}}\ \sum_{\left\{\alpha\in A_{jq}\middle|\alpha'-\widehat{l}_{jq}+1\le\alpha\le\alpha'\right\}}\ \sum_{\left\{\beta\in B_{jq}\middle|\beta'-\widehat{w}_{jq}+1\le\beta\le\beta'\right\}} \frac{\widehat{d}_{jq}}{a_{jq}}\cdot\pi_{\alpha\beta(\gamma'+h_{ik})}^{jquv} \\
\le&\sum_{\left\{\alpha\in A_{ik}\middle|\alpha'-\widehat{l}_{ik}+1\le\alpha\le\alpha'\right\}}\ \sum_{\left\{\beta\in B_{ik}\middle|\beta'-\widehat{w}_{ik}+1\le\beta\le\beta'\right\}} p_{ik}\cdot\pi_{\alpha\beta\gamma'}^{iktv} \\
+&\left(1-\sum_{\left\{\alpha\in A_{ik}\middle|\alpha'-\widehat{l}_{ik}+1\le\alpha\le\alpha'\right\}}\ \sum_{\left\{\beta\in B_{ik}\middle|\beta'-\widehat{w}_{ik}+1\le\beta\le\beta'\right\}}\pi_{\alpha\beta\gamma'}^{iktv}\right)\cdot M_4 \qquad (2.159)
\end{aligned}$$

$$\forall\ i\in N_c, k\in J_i^L, v\in V, t\in N\setminus\{n\}, \alpha'\in A_0, \beta'\in B_0, \gamma'\in \Gamma_0,$$

where

$$\widehat{d}_{ik}=d_{ik}+\sum_{j\in N_c}\sum_{q\in J_j^L}\left(\frac{supp_{jqik}}{a_{jq}}\cdot\widehat{d}_{jq}\right)\qquad\forall\ i\in N_c, k\in J_i^L; \qquad (2.160)$$

(2) backhaul:

$$\sum_{j\in N_c}\sum_{q\in J_j^B}\sum_{u\in N\setminus\{n\}}\sum_{\left\{\alpha\in A_{jq}\middle|\alpha'-\widehat{l}_{jq}+1\le\alpha\le\alpha'\right\}}\sum_{\left\{\beta\in B_{jq}\middle|\beta'-\widehat{w}_{jq}+1\le\beta\le\beta'\right\}}\frac{\widehat{d}_{jq}}{a_{jq}}\cdot\pi^{jquv}_{\alpha\beta(\gamma'+h_{ik})}$$
$$\le\sum_{\left\{\alpha\in A_{ik}\middle|\alpha'-\widehat{l}_{ik}+1\le\alpha\le\alpha'\right\}}\sum_{\left\{\beta\in B_{ik}\middle|\beta'-\widehat{w}_{ik}+1\le\beta\le\beta'\right\}}p_{ik}\cdot\pi^{iktv}_{\alpha\beta\gamma'}$$
$$+\left(1-\sum_{\left\{\alpha\in A_{ik}\middle|\alpha'-\widehat{l}_{ik}+1\le\alpha\le\alpha'\right\}}\sum_{\left\{\beta\in B_{ik}\middle|\beta'-\widehat{w}_{ik}+1\le\beta\le\beta'\right\}}\pi^{iktv}_{\alpha\beta\gamma'}\right)\cdot M_4 \tag{2.161}$$
$$\forall\ i\in N_c, k\in J_i^B, v\in V, t\in N\setminus\{n\}, \alpha'\in A_0, \beta'\in B_0, \gamma'\in \Gamma_0,$$

where

$$\widehat{d}_{ik}=d_{ik}+\sum_{j\in N_c}\sum_{q\in J_j^B}\left(\frac{supp_{jqik}}{a_{jq}}\cdot\widehat{d}_{jq}\right)\qquad\forall\ i\in N_c, k\in J_i^B. \tag{2.162}$$

For the linehaul items, $supp_{ikjq}$ $(i,j\in N_c, k\in J_i^L, q\in J_j^L)$ can be determined analogously to (2.63) or (2.64), and a_{ik} $(i\in N_c, k\in J_i^L)$ as in (2.139). For the backhaul items, $supp_{ikjq}$ $(i,j\in N_c, k\in J_i^B, q\in J_j^B)$ can be determined analogously to (2.63) or (2.107), and a_{ik} $(i\in N_c, k\in J_i^B)$ as in (2.140). The formulations for linehaul and backhaul items differ only in the usage of the item sets.

Variable domain constraints

The variable domain constraints (2.81)-(2.93) and (2.126)-(2.127) can be adopted unaltered from the models for the 3L-VRPTW and 3L-VRPCB(TW).

Chapter 3

Literature review

This chapter provides an overview of the relevant literature. Relevant VRP variants with backhauls and time windows are regarded (Chapter 3.1). Furthermore, a short overview of packing problems is provided (Chapter 3.2) as well as an overview of VRPs with loading constraints (Chapter 3.3).

3.1 Vehicle routing problems

The VRP was first introduced as the so-called "truck dispatching problem" by Dantzig and Ramser (1959) more than 50 years ago. The problem was presented as a generalization of the well-known travelling salesman problem. Ever since, the VRP has been studied intensively and many different variants of the problem have been identified. According to Irnich et al. (2014b), they can be classified based on their network characteristics, types of transportation requests, intra- and inter-route constraints, fleet characteristics and optimization objectives. Three problem variants, that are of interest in this thesis, are the VRP with time windows (VRPTW), VRP with backhauls (VRPB) and VRP with backhauls and time windows (VRPBTW).

3.1.1 Vehicle routing problems with time windows

A frequently investigated variant is the VRPTW. Whereas early works concentrate on conventional heuristic solution approaches (e.g. Pullen, 1967; Knight and Hofer, 1968; Madsen, 1976), the focus later shifts towards exact (e.g. Christofides et al., 1981; Kolen et al., 1987) and metaheuristic solution approaches (e.g. Garcia et al., 1994; Homberger and Gehring, 1999; Bent and van Hentenryck, 2004). Very good results are also obtained by the adaptive large neighbourhood search (ALNS) proposed by Pisinger and Ropke (2007) and the genetic algorithm (GA) presented by Vidal et al. (2013).

In the VRPTW literature, the minimization of the number of vehicles is often used as first objective, whereas distance minimization is usually the primary objective in other VRPs. According to Duhamel et al. (1997), distance minimization should be considered if the vehicles are not associated with any fixed, decision-relevant costs.

Focusing on the minimization of the number of tours would reduce the problem to a bin-packing problem. That is, the optimization would aim at "packing" customers into routes if the capacity constraint is not violated although they might be located far away from each other. Hence, this objective appears to be inappropriate for such VRPs. However,

> " [for] problems with time window constraints (or VRPs with maximal distance constraints), the bin-packing effect is alleviated. In this case, minimizing the number of routes looks as a natural objective. The time issues are addressed in the secondary objective, thus avoiding solutions with low distance travelled but large waiting times. " (Duhamel et al., 1997, p. 50)

Solomon (1987) introduces 56 VRPTW instances with 100 customers each, which are commonly used as benchmark instances. They are also used as benchmark instances in the course of this thesis and (partly) as a basis for the generation of 3L-VRPBTW instances. To date, 55 out of the 56 instances could be solved to proven optimality (concerning the minimization of the total travel distance). The last 20 instances have been solved rather recently (Jepsen et al., 2008; Desaulniers et al., 2008; Baldacci et al., 2011), which shows that there is still an interesting evolution happening in the field of exact solution approaches. Prominent large-scale instances with up to 1,000 customers are introduced by Gehring and Homberger (1999).

3.1.2 Vehicle routing problems with backhauls

The group of VRPBs contains further popular extensions of the VRP. Since such problem variants are one of the main aspects of this work, the relevant literature will be examined in more detail in the following section. Afterwards, an overview is given about the combination of the VRPTW and the VRPB, i.e. the VRPBTW.
Different variants of the VRPB can be classified. Since this work deals with the VRP with clustered backhauls (VRPCB), VRP with mixed backhauls (VRPMB) and VRP with simultaneous delivery and pickup (VRPSDP), the following literature review will focus on these problems as well.
The VRPB was first mentioned in the 1980s. In particular, the VRPCB (Deif and Bodin, 1984; Goetschalckx and Jacobs-Blecha, 1989) and the VRPMB (Golden et al., 1985; Casco et al., 1988) were studied at that time. Goetschalckx and Jacobs-

Blecha (1989) introduce instances with 25 to 100 customers that are widely accepted as standard instances in the VRPB literature.

Exact methods for the VRPCB are proposed by Toth and Vigo (1997) and Mingozzi et al. (1999). Both apply branch-and-bound (B&B) algorithms with different lower bounding procedures that could solve the majority of the instances of Goetschalckx and Jacobs-Blecha (1989) and Toth and Vigo (1996) with up to 100 customers to optimality. An exact approach for the VRPSDP is presented by Dell'Amico et al. (2006) who apply a branch-and-price (B&P) algorithm with an upper bounding heuristic based on dynamic programming and lower bounding based on state space relaxation.

Due to the complexity of the problem, the vast majority of researchers apply (conventional) heuristic and metaheuristic approaches for solving the different variants of the VRPB. Especially the early works focus on conventional heuristics. One of the first is presented by Deif and Bodin (1984), which is an extension of the savings algorithm of Clarke and Wright (1964). A penalty term is added to the savings of the edges connecting a linehaul with a backhaul customer in order to postpone the formation of mixed routes. The savings algorithm is also applied to the VRPMB by Golden et al. (1985) and Casco et al. (1988). Both use the heuristic to schedule the linehaul customers. Subsequently, the backhaul customers are added by means of different insertion heuristics.

Toth and Vigo (1996, 1999) present a cluster-first-route-second approach for the VRPCB. Beginning with a solution obtained by Lagrangian relaxation, linehaul and backhaul clusters are determined and matched. The routes for the clusters are then obtained by means of a modified farthest-insertion heuristic and improved by applying intra-route optimization. Further cluster-first-route-second approaches are, for example, applied by Min (1989) for the VRPSDP or by Halse (1992) for the VRPMB and VRPSDP. Other heuristic approaches include, but are not limited to, insertion heuristics (Salhi and Nagy, 1999; Dethloff, 2001), or partitioning heuristics (Anily, 1996; Mosheiov, 1998).

In later publications, the focus is shifted increasingly towards metaheuristic approaches. Osman and Wassan (2002) present a reactive tabu search (RTS) algorithm for the VRPCB. It generates initial solutions with saving-insertion and saving-assignment heuristics improved by 2-opt and 3-opt procedures. The neighbourhood is explored using interchange operators during the tabu search (TS). A mechanism

detecting recurrent solutions is used for controlling diversification and intensification of the search. Further (reactive) TS algorithms are proposed by Crispim and Brandão (2001), Brandão (2006) and Wassan (2007) for the VRPCB, by Wassan et al. (2008a) and Nagy et al. (2013) for the VRPMB, and by Tang Montané and Galvão (2006), Bianchessi and Righini (2007) and Wassan et al. (2008a,b) for the VRPSDP.

Wade and Salhi (2004) propose an ant colony optimization (ACO) approach for the VRPMB with special features regarding the selection of candidates, the size of the candidate list and the local trail update. Similar approaches are presented by Gajpal and Abad (2009) for the VRPCB, by Wassan et al. (2013) for the VRPMB and by Çatay (2010) for the VRPSDP. A more recent example of a metaheuristic is the variable neighbourhood search (VNS) by Polat et al. (2015) for the VRPSDP. The initial solution for the algorithm is generated using the savings algorithm. For this purpose, the enhancement of the savings formula of Altınel and Öncan (2005) is applied that also considers the customer demands. Eight different intra- and inter-route neighbourhood structures are employed during the search, such as 2- and 3-opt, swap or shift. Furthermore, a perturbation mechanism is used to escape local optima and each solution is further improved (if possible) by a variable neighbourhood descent (VND), which serves as a local search operator.

In addition to the routing constraints regarded in this thesis (see Chapter 2), further constraints and alternative problem features are considered in the literature. For example, Deif and Bodin (1984) and Tang Montané and Galvão (2006) restrict the route lengths. Multiple depots are taken into account by Min et al. (1992), Salhi and Nagy (1999) and Li et al. (2015), among others. Wade and Salhi (2002) restrict the mixture of linehaul and backhaul customers, i.e. the first backhaul customer in a route must not be visited before a certain number of linehaul customers was visited. In doing so, it should be avoided to visit a backhaul customer too early in a route which could cause high rearrangement efforts of the loaded goods. Nonetheless, the advantages of the VRPMB over the VRPCB should be exploited. Mitra (2005) considers a problem where the linehaul and/or backhaul demand of a customer may exceed the vehicle capacity. Hence, a customer can be visited more than once.

Table 3.1 provides an overview of the VRPB literature. It contains information about the considered VRPB variants (column “prob.”) (i.e. the VRPCB (C), VRPMB (M) or VRPSDP (S)), the nature of the solution approaches (i.e. exact (E),

metaheuristic (Mh) or conventional heuristic (H); "appr."), the applied algorithms ("algorithm") and the benchmark instances used for testing ("instances").

Table 3.1: Literature overview VRPB

reference	**prob.**	**appr.**	**algorithm***	**instances****
Deif and Bodin (1984)	CB	H	Sav	**DB84**
Golden et al. (1985)	M	H	Sav + IH	**GBAS85**
Casco et al. (1988)	M	H	Sav + IH	**CGW88**
Goetschalckx and Jacobs-Blecha (1989)	C	H	space-filling curve, 2-opt, 3-opt	**GJB89**
Min (1989)	S	H	C1R2	**Min89**
Halse (1992)	M, S	H	C1R2	GBAS85, GJB89, Min89, **Hal92**
Min et al. (1992)	C	H	C1R2	**Min92**
Goetschalckx and Jacobs-Blecha (1993)	C	H	general assignment	GJB89
Anily (1996)	C	H	circular partitioning	-
Toth and Vigo (1996)	C	H	C1R2	GJB89, **TV96**
Toth and Vigo (1997)	C	E	B&B	GJB89, TV96
Mosheiov (1998)	M	H	tour partitioning	**Mos98**
Mingozzi et al. (1999)	C	E	variable reduction, LP	GJB89, TV96
Salhi and Nagy (1999)	M, S	H	cluster IH	**SN99a**, **SN99b**
Toth and Vigo (1999)	C	H	C1R2	GJB89, TV96
Crispim and Brandão (2001)	C	Mh	RTS, VNS	TPS96
Dethloff (2001)	S	H	IH	Min89, SN99b, **Det01**
Dethloff (2002)	M	H	IH	SN99a
Osman and Wassan (2002)	C	Mh	RTS	GJB89, TV96
Wade and Salhi (2002)	M	H	IH	GJB89, TV97
Wade and Salhi (2004)	M	Mh	ACO	GJB89
Crispim and Brandão (2005)	M, S	Mh	TS×VND	SN99a, SN99b

Table 3.1: Literature overview VRPB (*continued*)

reference	prob.	appr.	algorithm*	instances**
Mitra (2005)	S	E, H	MILP, route construction	**M05**
Nagy and Salhi (2005)	M, S	H	four-phase-heuristic	SN99a, SN99b
Brandão (2006)	C	Mh	TS	GJB89, TV96
Chen and Wu (2006)	S	Mh	IH	SN99b, **CW06**
Dell'Amico et al. (2006)	S	E	B&P	**DRS06**
Ghaziri and Osman (2006)	C	Mh	neural network	TV96
Ropke and Pisinger (2006a)	C, M, S	Mh	LNS	GJB89, TV96, SN99a, Min89, SN99b, Det01
Tang Montané and Galvão (2006)	S	Mh	TS	Min89, SN99b, Det01, **TMG06**
Bianchessi and Righini (2007)	S	Mh	LS, TS	**BR07**
Ganesh and Narendran (2007)	C	Mh	GA	GJB89, TV96
Wassan (2007)	C	Mh	RTS	GJB89, TV96
Wassan et al. (2008a)	M, S	Mh	RTS	GJB89, TV97, SN99a, Det01, TG06
Wassan et al. (2008b)	S	Mh	RTS	SN99b
Gajpal and Abad (2009)	C	Mh	ACO	GJB89, TV97
Tütüncü et al. (2009)	C, M	Mh	visual interactive	GJB89, TV96
Zachariadis et al. (2009)	S	Mh	TS×GLS	SN99b, Det01, TMG06
Çatay (2010)	S	Mh	ACO	Min89, Det01, SN99b
Subramanian et al. (2010a)	S	Mh	VNS×ILS	Det01, SN99b, TMG06
Subramanian et al. (2010b)	S	E	B&C	Det01, SN99b, TMG06
Zachariadis et al. (2010)	S	Mh	adaptive memory algorithm	Det01, SN99b, TMG06
Hezer and Kara (2011)	S	Mh	bacterial foraging	Det01
Maquera et al. (2012)	S	Mh	SS	SN99b, Det01, TMG06

Table 3.1: Literature overview VRPB (*continued*)

reference	**prob.**	**appr.**	**algorithm***	**instances****
Jun and Kim (2012)	S	Mh	perturbation based algorithm	SN99b
Tasan and Gen (2012)	S	Mh	GA	**TG12**
Zachariadis and Kiranoudis (2012)	C	Mh	LS	GJB89
Zhang et al. (2012)	S	Mh	SS, GA	Det01, **ZCZ12**
Goksal et al. (2013)	S	Mh	PSO	Det01, SN99b
Nagy et al. (2013)	M	Mh	RTS	GJB89, TV97, SN99a
Rieck and Zimmermann (2013)	S	E	MILP	**RZ13**, M05, CW06, DRS06, Det01, SN99b
Subramanian et al. (2013)	S	E	BC&P	SN99b, Det01, TMG06
Wassan et al. (2013)	M	Mh	ACO	GJB89, TV97, SN99a
Palhazi Cuervo et al. (2014)	C	Mh	ILS	GJB89, TV97
Avci and Topaloglu (2015)	M, S	Mh	adaptive LS	Det01, SN99b
Belloso et al. (2015)	C	Mh	randomization heuristic	GJB89
García-Nájera et al. (2015)	C, M, S	Mh	EA	GJB89, SN99a, SN99b
Li et al. (2015)	S	Mh	ILS	SN99b
Polat et al. (2015)	S	Mh	VNS	SN99b, Nagy et al. (2015), **PKKG15**
Avci and Topaloglu (2016)	S	Mh	LS×TS	**AT16**

ACO: ant colony optimization, appr.: approach, B&B: branch-and-bound, B&C: branch-and-cut, BC&P: branch-cut-and-price, B&P: branch-and-price, C: VRPCB, C1R2: cluster-first-route-second, E: exact, EA: evolutionary algorithm, GA: genetic algorithm, GLS: guided local search, H: heuristic (conventional), ILS: iterated local search, IH: insertion heuristic, LNS: large neighbourhood search, LP: linear programming, LS: local search, M: VRPMB, Mh: metaheuristic, MILP: mixed integer LP, prob.: problem variant, PSO: particle swarm optimization, S: VRPSDP, Sav: savings heuristic, SS: scatter search, RTS: reactive TS, TS: tabu search, VND: variable neighbourhood descent, VNS: variable neighbourhood search; *"... × ...": hybrid of the two mentioned algorithms; **written in bold: first introduction of the instance set (see also Table 3.3)

3.1.3 Vehicle routing problems with backhauls and time windows

Similarly to the VRPB, there are a few exact and (conventional) heuristic approaches for the VRPBTW. The majority of solution methods represent metaheuristics. The problem with clustered backhauls is the variant that is most often considered in connection with time windows. One of the first exact approaches is the B&B algorithm of Yano et al. (1987) for the VRPCB with time windows (VRPCBTW). The authors present an application for a retail chain and apply the algorithm to instances with up to 50 stores (linehaul) and 200 suppliers (backhaul). However, the number of linehaul and backhaul customers is limited to four of each per route. Furthermore, capacity constraints regarding weight and volume, maximum driving times and maximum time on the road (each truck may be used by more than one driver) as well as opening and closing times at the destinations are considered. Gélinas et al. (1995) also present a B&B algorithm for the VRPCBTW and introduce instances that are often referred to in the subsequent literature on the problem.

Apart from two conventional heuristic approaches (Derigs and Metz, 1992; Thangiah et al., 1996), various metaheuristics are introduced for the VRPCBTW. One of the first is the approach of Potvin et al. (1996) who propose a route construction heuristic combined with a GA. The route construction heuristic is a greedy heuristic that inserts customers in positions that minimize detours and service delays. The GA is applied in order to determine the sequence in which the customers are inserted.

Duhamel et al. (1997) propose a TS algorithm with a relatively short tabu list that stores information from the last five iterations and three different neighbourhoods, namely 2-opt*, Or-opt and swap. One of them is randomly chosen at each iteration.

In a more recent work, Ong and Suprayogi (2011) investigate the VRPCBTW with multiple trips, i.e. each vehicle can perform more than one tour during the planning period. The authors present an ACO algorithm with the initial solution being generated via sequential insertion.

The more complex problem variants VRPMB with time windows (VRPMBTW) and VRPSDP with time windows (VRPSDPTW) have not been tackled very often yet. An exact approach for the VRPSDPTW is presented by Angelelli and Mansini (2002): a B&P algorithm based on a set covering formulation. The approach can solve instances with up to 20 customers to optimality. Apart from that, the VRPMBTW and VRPSDPTW are mainly solved using metaheuristics.

Kontoravdis and Bard (1995) present a greedy randomized adaptive search procedure (GRASP) for the VRPMBTW. In the initialization phase, customers are assigned to routes using parallel insertion. The seed customers for the initialization are selected taking into account both their distance from the depot and their time window width. Subsequently, the GRASP is applied in order to find and improve feasible solutions.

Moreover, local search-based approaches are proposed, for example, simulated annealing (SA) (Hasama et al., 1998) or TS (Fan, 2011). Further population-based metaheuristics are, e.g., ACO (Reimann and Ulrich, 2006) or particle swarm optimization (PSO) (Belmecheri et al., 2013). SA and guided local search (GLS) approaches for both the VRPCBTW and the VRPMBTW are presented by Hasama et al. (1998) and by Zhong and Cole (2005), respectively. Furthermore, Ropke and Pisinger (2006a) develop an ALNS for a variety of problem variants. Of the problems discussed in this thesis, the VRPCB, VRPMB, VRPSDP, VRPCBTW and VRPMBTW are covered.

Table 3.2 provides an overview of the literature combing the VRPB and the VRPTW. In addition to the information given in Table 3.1, Table 3.2 also states the optimization objective (column "obj.").

Table 3.2: Literature overview VRPBTW

reference	**prob.**	**obj.**	**appr.**	**algorithm***	**instances****
Yano et al. (1987)	C	D	E	B&B	Retail case study
Derigs and Metz (1992)	C	D	H	matching	**DM92**
Gélinas et al. (1995)	C	D	E	B&B	**GDDS95**
Kontoravdis and Bard (1995)	M	VD	Mh	GRASP	**KB95**
Potvin et al. (1996)	C	D	Mh	GA	GDDS95
Thangiah et al. (1996)	C	VD	H	insertion	GDDS95, **TPS96**
Duhamel et al. (1997)	C	D, VD	Mh	TS	GDDS95
Hasama et al. (1998)	C, M	VD	Mh	SA	TPS96
Angelelli and Mansini (2002)	S	D	E	B&P	**AM02**
Reimann et al. (2002)	C	VD	Mh	ACO	GDDS95
Zhong and Cole (2005)	C, M	VD	Mh	GLS	GDDS95, KB95

Table 3.2: Literature overview VRPBTW (*continued*)

reference	prob.	obj.	appr.	algorithm*	instances**
Reimann and Ulrich (2006)	M	VD	Mh	ACO	GDDS95
Ropke and Pisinger (2006a)	C, M	VD	Mh	ALNS	GDDS95, TPS96, KB95
Fan (2011)	S	D	Mh	TS	**Fan11**
Ong and Suprayogi (2011)	C	VD	Mh	ACO	**OS11**
Wang and Chen (2012)	S	VD	Mh	GA	**WC12**
Belmecheri et al. (2013)	M	D	Mh	PSO	**BPYA13**
Zhang et al. (2014)	S	D	Mh	ACO×TS	**ZCZ14**
Wang et al. (2015)	S	VD	Mh	SA	WC12, **WMZS15**

ALNS: adaptive large neighbourhood search, appr.: approach, C: VRPCB, D: minimize distance, E: exact, H: heuristic, M: VRPMB, Mh: metaheuristic, obj.: objective, prob.: problem variant, S: VRPSDP, SA: simulated annealing, VD: minimize number of used vehicles (first objective) + minimize distance (second objective); *"... ×...": hybrid of the two mentioned algorithms; **written in bold: first introduction of the instance set (see also Table 3.3). Further abbreviations of solution approaches are provided in Table 3.1

Hereinafter, let VRPs with and without time windows and backhauls be collectively referred to as VRPBTWs. Table 3.3 provides an overview of VRPBTW instances from the literature. Only instances referred to in this thesis are listed. Thus, the table is without any claim to completeness regarding all available instances. The respective references and problem variants are listed as well as the characteristics (e.g. number of customers (n) or ratio of linehaul and backhaul customers) and the numbers of provided instances (#). Moreover, the table contains information about previously published instances an instance set is based on (basis), if necessary.

Table 3.3: VRP instances

reference	abbrev.	problem	characteristics	#	basis
Christofides and Eilon (1969)	CE69	CVRP	n: 50-100	3	-
Christofides et al. (1979)	CMT79	CVRP	n: 50-199	14	-

Table 3.3: VRPBTW instance sets (*continued*)

reference	**abbrev.**	**problem**	**characteristics**	**#**	**basis**
Deif and Bodin (1984)	DB84	VRPCB	n: 100-300; %BH: 10-50	n.a.	n.a.
Golden et al. (1985)	GBAS85	VRPMB	n: 50; %BH: 10	1	CE69
Solomon (1987)	Sol87	VRPTW	n: 100	56	-
Casco et al. (1988)	CGW88	VRPMB	n: 61; %BH: 18	n.a.	n.a.
Goetschalckx and Jacobs-Blecha (1989)	GJB89	VRPCB	n: 25-150; %BH: 20, 33, 50	62	-
Min (1989)	Min89	VRPSDP	n: 22	1	real world
Derigs and Metz (1992)	DM92	VRPCBTW	n: 28-160	4	real world
Halse (1992)	Hal92	VRPSDP	n: 22-150	n.a.	n.a.
Min et al. (1992)	MCS92	VRPCB	n: 161	n.a.	n.a.
Gélinas et al. (1995)	GDDS95	VRPCBTW	n: 100; %BH: 10, 30, 50	15	Sol87
Kontoravdis and Bard (1995)	KB95	VRPMBTW	n: 100; %BH: 50	27	Sol87
Thangiah et al. (1996)	TPS96	VRPCBTW	n: 250-500; %BH: 10, 30, 50	24	Sol87
Toth and Vigo (1996)	TV96	VRPCB	n: 21-100; %BH: 20, 33, 50	33	TSPLIB
Mosheiov (1998)	Mos98	VRPMB	n: 50, 100; %BH: 50	300	-
Gehring and Homberger (1999)	GH99	VRPTW	n: 200-1,000	300	Sol87
Salhi and Nagy (1999)	SN99a	VRPMB	n: 50-199; %BH: 10, 25, 50	42	CMT79
Salhi and Nagy (1999)	SN99b	VRPSDP	n: 50-199	28	CMT79
Dethloff (2001)	Det01	VRPSDP	n: 50	40	-
Angelelli and Mansini (2002)	AM02	VRPSDPTW	n: 20	56	Sol87

Table 3.3: VRPBTW instance sets (*continued*)

reference	**abbrev.**	**problem**	**characteristics**	**#**	**basis**
Mitra (2005)	M05	VRPSDP	n: 19	110	-
Dell'Amico et al. (2006)	DRS06	VRPSDP	n: 20, 40	120	Sol87, TV69, real world
Chen and Wu (2006)	CW06	VRPSDP	n: 15, 17, 20	25	Sol87, GH99
Tang Montané and Galvão (2006)	TMG06	VRPSDP	n: 100-400	18	Sol87, GH99
Bianchessi and Righini (2007)	BR07	VRPSDP	n: 50, 100; %BH: 50	300	-
Fan (2011)	Fan11	VRPSDPTW	n: 10-50	6	-
Ong and Suprayogi (2011)	OS11	VRPCBTW	n: 100; %BH: 50	6	-
Tasan and Gen (2012)	TG12	VRPSDP	n: 15-44	24	TSPLIB
Wang and Chen (2012)	WC12	VRPSDPTW	n: 10, 25, 50, 100	65	Sol87
Zhang et al. (2012)	ZCZ12	VRPSDP	n: 200, 400	60	-
Belmecheri et al. (2013)	BPYA13	VRPMBTW	n: 5-100; %BH: 20	168	Sol87
Rieck and Zimmermann (2013)	RZ13	VRPSDP	n: 20-60	100	-
Zhang et al. (2014)	ZCZ14	VRPSDPTW	n: 100	56	Sol87
Polat et al. (2015)	PKKG15	VRPSDP	n: 10-40	10	-
Wang et al. (2015)	WMZS15	VRPSDPTW	n: 200-1,000	30	GH99
Avci and Topaloglu (2016)	AT16	VRPSDP	n: 10-550	28	-

%BH: share of backhaul customers (in the case of VRPCB(TW) and VRPMB(TW)), #: number of instances, n: number of customers, n.a.: not available, TSPLIB: VRP instances available in the TSPLIB (cf. Reinelt, 1991)

3.2 Packing problems

Cutting and packing (C&P) problems have gained increasing research interest in the past decades. Being generally NP-hard, they constitute challenging academic problems but they are also difficult to solve in practice (Pisinger, 2002). Due to the large variety of practical applications, numerous problem variants have been studied that can be categorized, e.g., based on the typology proposed by Wäscher et al. (2007).

In general, C&P problems can be described as follows: subsets of small items have to be assigned to one or more large objects (containers) such that they are placed completely within the large objects and do not overlap (Wäscher et al., 2007). The items and containers can be one-, two- or three-dimensional, or of even larger (non-geometric) dimensions. As the focus of this thesis is on the loading of 3D items, only 3D packing problems are regarded in the remainder of this subsection. Furthermore, an objective function is usually given that is to be optimized. The objective of an optimization problem can be the minimization of input. This includes, for example, the minimization of the utilized container length or height (open dimension problem), or the number of used containers (e.g. bin packing problems). In this case, a set of items has to be assigned to a set of containers and all items have to be placed. Furthermore, the objective can consist in maximizing the output, i.e. the packed value or volume (e.g. knapsack problem). In this case, not all items can be accommodated by the available container(s) and a selection of the assigned items is required. For the three-dimensional case, the optimization problems are referred to as container loading problems (CLPs) (e.g. Wäscher et al., 2007; Bortfeldt and Wäscher, 2013).

C&P problems can be classified (among others) based on the optimization objective (see above), the assortment of the items and containers, and the heterogeneity of the items (Wäscher et al., 2007). Moreover, different practically relevant constraints are considered in the literature. According to Bortfeldt and Wäscher (2013), those constraints can refer to the containers (e.g. weight limits or weight distribution constraints), the individual items to be packed (e.g. orientation or stacking constraints), sets of items (e.g. complete-shipment or separation constraints), the relationship between the containers and the items (e.g. relative positioning or multi-drop constraints), or the final load (e.g. stability or complexity constraints).

Solving the CLP to optimality is difficult due to its complexity. Thus, only very few exact solution methods exist and only few models have been proposed. Chen et al. (1995) introduce a mixed integer linear programming (MILP) formulation, which is later extended by Fasano (1999) and Padberg (2000). Fekete et al. (2007) (see also Fekete and Schepers, 1997, 2004) present an exact two-level tree search algorithm. At the first level, a higher-dimensional knapsack problem consisting of the search for a subset of items is solved. In order to determine whether or not a feasible packing plan exists for the subset, an orthogonal packing problem (OPP) (see below) is solved at the second level.

Due to the difficulty of the problems, numerous heuristic solution approaches are proposed for solving the CLP, including conventional heuristics (e.g. construction and improvement heuristics), metaheuristics and heuristic tree search approaches. They are usually based on packing approaches that determine geometrical structures of packing plans (Fanslau and Bortfeldt, 2010). These approaches include, but are not limited to, wall building, stack building, layer building and block building approaches (cf. Pisinger, 2002; Fanslau and Bortfeldt, 2010). *Wall-building* approaches fill the container with vertical layers of items (parallel to the $W \times H$-plane) along the length axis. Such approaches are presented, among others, by George and Robinson (1980), Gehring et al. (1990), Bortfeldt and Gehring (2001) and Pisinger (2002). *Stack building* algorithms (e.g. Gilmore and Gomory, 1965; Bischoff and Ratcliff, 1995; Gehring and Bortfeldt, 1997) arrange stacks of items on the container floor. In the case of *horizontal layer building* approaches, the container is filled from bottom to top by horizontal layers. The approach is used, e.g., by Bischoff et al. (1995) and Terno et al. (2000). *Block building* approaches form blocks containing items that are usually identical and have the same spatial orientation. Several blocks are then arranged within the container (e.g. Eley, 2002; Bortfeldt et al., 2003). Recently, block building approaches have been proposed that do not demand that the items that form a block are of the same type or have the same spatial orientation (e.g. Fanslau and Bortfeldt, 2010; Araya and Riff, 2014).

Examples for the mentioned packing approaches are illustrated in Figure 3.1.

In order to obtain solutions of higher quality, the construction heuristics are usually integrated into more sophisticated search frameworks. For example, Gehring and Bortfeldt (1997) present a solution approach consisting of a stack-building algorithm for generating stacks of items and a GA which is responsible for solving the resulting

problem of covering the container floor with the item stacks. In other GAs, e.g. applied by Hemminki (1994) or Bortfeldt and Gehring (2001), vertical layers are formed by a wall-buiding procedure and GA operators transfer layers from parent to offspring solutions. Further variants of GAs are proposed by Gonçalves and Resende (2012) and Zheng et al. (2015). Other metaheuristic approaches include, for example, GRASP (Moura and Oliveira, 2005; Parreño et al., 2008), TS (Bortfeldt et al., 2003; Mack et al., 2004; Allen et al., 2011), or SA (Sixt, 1996; Mack et al., 2004).

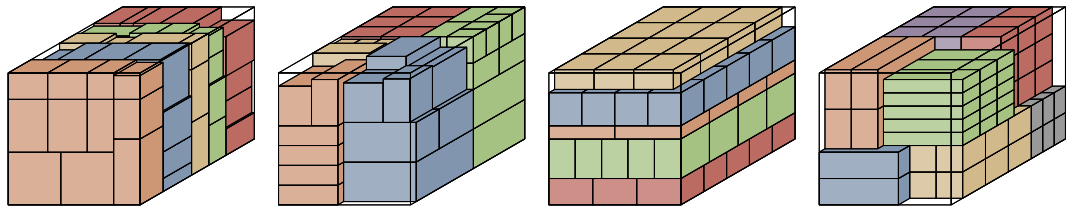


Figure 3.1: Heuristic approaches for solving the CLP

Heuristic tree search approaches are considered by Terno et al. (2000), Eley (2002), Hifi (2002), Pisinger (2002) and Fanslau and Bortfeldt (2010). Furthermore, Araya and Riff (2014) present a beam search algorithm, which is a variant of the B&B approach. However, it does not guarantee optimality. The beam search heuristic searches a graph by expanding only the most promising nodes at each level.

A recent review and comparison of solution approaches for the CLP is provided by Zhao et al. (2016).

In contrast to the packing optimization problems, the objective of a packing satisfiability (or decision) problem is not to maximize an output or minimize an input, but to find out whether or not a feasible packing plan exists that accommodates all given items in a given container. Such a problem is referred to as an orthogonal packing problem (OPP) (e.g. Fekete and Schepers, 2004).

The focus in the 3D C&P literature lies on the CLP and only very few publications exist that deal with the OPP in particular. The OPP is – to the best of the author's knowledge – first mentioned by George and Robinson (1980) who propose a wall-building heuristic for solving the problem. It is further considered by Fekete et al. (2007) (see also Fekete and Schepers, 1997, 2004) as a subproblem in a solution algorithm for the CLP (see above).

3.3 Vehicle routing problems with multi-dimensional loading constraints

The focus of this section will be on 3L-VRPs. VRPs with two-dimensional loading constraints (2L-VRPs) will only be mentioned briefly. VRPs with one-dimensional loading constraints are not regarded.

The 3L-CVRP was introduced first by Gendreau et al. (2006). Regarding the packing subproblem, the authors apply four constraints (in addition to the standard geometrical constraints): fixed vertical orientation, fragility, vertical stability and LIFO policy (see Chapter 2.1.2). These constraints are also used in the majority of papers on 3L-VRPs, that were published later. Moreover, Gendreau et al. (2006) present the first benchmark instances for the 3L-CVRP based on well-known CVRP instances (cf. Table 3.5).

Some authors take different or additional loading constraints into account. Tarantilis et al. (2009) and Ceschia et al. (2013) consider a variation of the LIFO constraint, which assumes manual reloading instead of loading and unloading with a forklift, i.e. the items do not need to be lifted. Hence, an item that is unloaded later in the route can be placed above (but not directly on top) of an item that is unloaded earlier. Ceschia et al. (2013) introduce the reachability, load bearing and robust stability constraints to the 3L-CVRP. Pace et al. (2015) present the transportation of fibre boards, which can be packed into stacks. Loading the stacks, a balanced distribution of the load (with respect to the weight) must be guaranteed.

Moura (2008) and Moura and Oliveira (2009) are the first to present the 3L-VRPTW. The latter also give a mathematical formulation of the combined routing and packing problem. As it is usual for routing problems with time windows, the authors also regard the minimization of the vehicle fleet as an objective and Moura (2008) even considers the maximization of the utilized volume. Furthermore, the authors regard horizontal stability, i.e. at least three sides of an item have to be supported by other items or the container side walls.

In combination with the 3L-VRP, only very few variants of pickup and delivery problems have been considered yet. Bortfeldt et al. (2015) investigate the 3L-VRPCB. Reil et al. (2017) study different variants of the 3L-VRPBTW, namely the 3L-VRPTW, 3L-VRPCBTW, 3L-VRPMBTW, 3L-VRPSDPTW and the 3L-VRP with divisible delivery and pickup and time windows (3L-VRPDDPTW). Whereas only

the rear loading approach is applied to the 3L-VRPTW and 3L-VRPCBTW, the remaining problem variants are solved under consideration of rear loading with reloading, side loading and vertical and horizontal loading space partition. The classical pickup and delivery problem with three-dimensional loading constraints (3L-PDP) – dealing with the transportation of items from pickup to delivery locations – is presented by Bartók and Imreh (2011) and Männel and Bortfeldt (2016).

3L-VRPs are complex optimization problems requiring a solution for both a routing and several packing problems. Junqueira et al. (2013) have attempted to solve a 3L-CVRP by means of standard optimization software. As the instances that could be solved are very small (3-8 customers and 32 items), this approach is not suitable for practically relevant instance sizes. The complexity of the integrated routing and packing problem calls for heuristic solution approaches. The routing problem is often solved using a metaheuristic approach. For example, Gendreau et al. (2006) apply a TS algorithm to the routing problem. The neighbourhood is explored by moving customers between the routes. The affected routes are subsequently re-optimized using a 4-opt generalized insertion procedure. Further TS procedures are presented by Wang et al. (2010) and Wisniewski et al. (2011). Miao et al. (2012) present a GA. Variants of the VNS are utilized by Wei et al. (2014) and Bortfeldt et al. (2015). In addition, Bortfeldt et al. (2015) consider the ALNS presented by Ropke and Pisinger (2006a,b) for one-dimensional VRPs as a second, alternative procedure for the routing problem.

On the contrary, rather simple construction heuristics are usually employed in order to test whether a feasible packing plan can be found for the items of the customers assigned to a route. These are mostly variants of the bottom-left heuristic and the touching area heuristic.[13] More advanced packing procedures are, for example, the local search algorithm of Zhang et al. (2015)[14] or the tree search algorithm of Bortfeldt (2012).

Whereas most solution approaches constitute algorithms that solve the routing and packing problems in an integrated way, Bortfeldt and Homberger (2013) propose a different approach: The packing problem is solved first, the routing problem is solved afterwards. That is, a 3D strip packing problem is solved for each customer so that a strip of items of height H, width W and with minimum length is provided

[13] The heuristics are explained in greater detail in Chapter 4.1.

[14] The procedure is described in Chapter 4.1.3

for each customer. Consequently, loading length constraints constitute the capacity constraints of the resulting routing problem. This approach is further extended by Reil et al. (2017) for solving 3L-VRPBTWs.

In addition, VRPBs have been studied under consideration of 2D loading constraints. Dominguez et al. (2015) present a hybrid algorithm combining large neighbourhood search (LNS) and biased randomization for a 2L-VRP with clustered backhauls (2L-VRPCB). A 2L-VRP with mixed backhauls (2L-VRPMB) is considered by Pinto et al. (2015), which is solved by means of an insertion heuristic. Moreover, Zachariadis et al. (2016) regard a 2L-VRP with simultaneous delivery and pickup (2L-VRPSDP). The simultaneous transport of linehaul and backhaul items is approached with a (vertical) loading space separation. The applied packing heuristic also considers to utilize the backhaul section for linehaul items that are delivered before the first backhaul item is picked up. Vice versa, the linehaul section can be used for backhaul items after the last linehaul item was unloaded.

Iori and Martello (2010) and Pollaris et al. (2015) provide detailed overviews of the literature regarding 2L-VRPs and 3L-VRPs. Moreover, Côté et al. (2017) examine the value of considering routing and loading problem in an integrated way. Applying an exact integrated solution algorithm and different non-integrated approaches to instances for the CVRP with two-dimensional loading constraints (2L-CVRP), the authors conclude that non-integrated approaches can result in considerable cost increases of (on average) about 6-9 %. In addition, the number of required vehicles could be decreased by applying an integrated approach.

An overview of the literature dealing with 3L-VRP is presented in Table 3.4. The packing constraints considered in the respective papers (columns “WL” to “R”) and the applied optimization criteria (“RC” to “VU”) are indicated. In the columns “routing” and “packing”, the solution approaches for the respective subproblem are shortly listed. The last column contains the used benchmark instances.

Table 3.5 provides an overview of 3L-VRPBTW and CLP instance sets used in the literature referred to above. Thus, the table is without any claim to completeness regarding all available instances. The respective references and problem variants are listed as well as the characteristics (e.g. number of customers (n) or total number of items (m)) and the numbers of provided instances (#). Moreover, the table contains information about previously published instances an instance set is based on (basis), if available.

Table 3.4: Literature overview 3L-VRP

reference	problem	packing constraints									objective			solution approach		instances**
		WL	VS	HS	LB	F	HR	VR	LF	R	RC	V	VU	routing*	packing	
Gendreau et al. (2006)	3L-CVRP	x	x			x	x		x		x			TS	TS with CH (BL + TA)	**GILM06**, RW
Aprile et al. (2007)	3L-CVRP								x		x			SA	strip packing approach	**AEGLP07**
Attanasio et al. (2007)	3L-CVRP	x	x				x		x			x		cutting plane	TS with CH	RW
Moura (2008)	3L-VRPTW		x	x			x	x	x		x	x	x	GA	GRASP	MO09[15]
Moura and Oliveira (2009)	3L-VRPTW		x	x			x	x	x		x	x		CH	GRASP	**MO09**
Tarantilis et al. (2009)	3L-CVRP	x	x			x	x		x		x			TS×GLS	various CH	GILM06, **TZK09**
Fuellerer et al. (2010)	3L-CVRP	x	x			x	x		x		x			ACO	various CH	GILM06
Wang et al. (2010)	3L-CVRP	x	x			x	x		x		x			TS	LS with DBLF + TA	GILM06
Bartók and Imreh (2011)	3L-PDP	x									x			LS	block algorithm	**BI11**
Ma et al. (2011)	3L-CVRP	x	x			x	x		x					TS	LS	GILM06
Wisniewski et al. (2011)	3L-CVRP	x	x			x	x		x		x			TS	BL	GILM06
Bortfeldt (2012)	3L-CVRP	x	x			x	x		x		x			TS	TrS	GILM06, TZK09
Massen et al. (2012)	3L-CVRP	x	x			x	x		x					column generation	TrS	GILM06
Miao et al. (2012)	3L-CVRP	x	x			x	x		x		x	x		GA	TS	GILM06
Zachariadis et al. (2012)	PPVRPTW		x				x				x			LS	CH	**ZTK12**, RW
Zhu et al. (2012)	3L-CVRP	x	x			x	x		x		x			TS	LS with DBLF + TA	GILM06, TZK09
Bortfeldt and Homberger (2013)	3L-VRPTW	x	x			x	x	x	x		x	x		(μ, λ)-ES×TS	TS	GILM06, MO09, **BH13**
Ceschia et al. (2013)	3L-CVRP	x	x		x	x	x		x	x	x			SA×LNS	various CH	GILM06, **CSS13**
Junqueira et al. (2013)	3L-CVRP		x		x	x			x	x	x			exact (GUROBI)		**JOCM13**
Lacomme et al. (2013)	3L-CVRP	x					x				x			GRASP×evol. LS	2-stage procedure	GILM06, **LTD13**
Ruan et al. (2013)	3L-CVRP	x	x			x					x	x		BCA	various CH	GILM06
Wei et al. (2014)	3L-HFVRP	x	x				x		x		x	x		AVNS	first fit	GILM06, TZK09, **WTL14**
Bortfeldt et al. (2015)	3L-VRPCB	x	x			x	x		x		x			ALNS, VNS	TrS	**BHMM15**

[15] The instances were first introduced in a working paper in 2007 on which the publication Moura and Oliveira (2009) is based.

Table 3.4: Literature overview 3L-VRP (*continued*)

		packing constraints									objective			solution approach		
reference	**problem**	WL	VS	HS	LB	F	HR	VR	LF	R	RC	V	VU	routing*	packing	**instances****
Pace et al. (2015)	3L-HFVRPTW	x	x						x		x			ILS, SA	specialized heuristic for grouping boards and placing stacks	**PTMA15**
Tao and Wang (2015)	3L-CVRP	x	x			x	x		x		x			TS	least waste heuristic, TA	GILM06, TZK09
Zhang et al. (2015)	3L-CVRP	x	x			x	x		x					evolutionary LS	LS with open space heuristic	GILM06, TZK09
Escobar-Falcón et al. (2016)	3L-CVRP	x	x			x	x		x		x			exact (branch-and-cut)	GRASP	GILM06
Männel and Bortfeldt (2016)	3L-PDP	x	x			x	x		x		x			ALNS	TrS	**MB16**
Reil et al. (2017)	various	x	x			x	x	x	x		x	x		(μ, λ)-ES×TS	TS	GILM06, MO09, BH13, **RBM17**
Zhang et al. (2017)	3L-CVRP	x	x			x	x		x		x			TS×BCA	DBLF	GILM06, Sol87

3L-VRPHF(TW): 3L-VRP with a heterogeneous fleet (and time windows), ACO: ant colony optimization, (A)LNS: (adaptive) large neighbourhood search, (A)VNS: (adaptive) variable neighbourhood search, BCA: bee colony approach, BL: bottom-left, CH: construction heuristic, DBLF: deepest-bottom-left-fill, F: fragility, GA: genetic algorithm, GLS: guided local search, HR: horizontal rotation, HS: horizontal stability, ILS: iterated local search, LB: load-bearing strength, LF: LIFO, LS: Local search, PPVRPTW: pallet-packing VRP with time windows, R: reachability, RC: min. routing cost, RW: real-world instances (no further specification), SA: simulated annealing, TA: touching area, TrS: tree search, TS: tabu search, V: min. number of vehicles, VR: vertical rotation, VS: vertical stability, VU: max. volume utilization, WL: weight limit; (μ, λ)-ES: (μ, λ)-evolution strategy, *"$\ldots \times \ldots$": hybrid of the two mentioned algorithms, **written in bold: first introduction of the instance set (see also Table 3.5)

Table 3.5: 3L-VRPBTW and CLP instance sets

reference	**abbrev.**	**problem**	**characteristics**	**#**	**basis**
Bischoff and Ratcliff (1995)	BR95	CLP	3-100 types	1,400	-
Gendreau et al. (2006)	GILM06	3L-CVRP	n: 15-100; m: 32-198	27	TSPLIB
Aprile et al. (2007)	AEGLP07	3L-CVRP	n: 10-100; m: n.a.	50	-
Moura and Oliveira (2009)	MO09	3L-VRPTW	n: 25; m: 1,050-1,550	46	Sol87, BR95
Tarantilis et al. (2009)	TZK09	3L-CVRP	n: 50-12; m: 73-379	12	-
Bartók and Imreh (2011)	BI11	3L-PDP	n: 19; m: 100-5,000	n.a.	-
Bortfeldt and Homberger (2013)	BH13	3L-VRPTW	n: 100-1,000; m: 5,000-50,000	120	Sol87, GH99, BR95
Zachariadis et al. (2012)	ZTK12	PPVRPTW	n: 50-199; m: 444-2,924	138	CMT79, Sol87
Ceschia et al. (2013)	CSS13	3L-CVRP	n: 11-129; m: 254-8,060	13	-
Junqueira et al. (2013)	JOCM13	3L-CVRP	n: 3-8; m: 32	6	-
Lacomme et al. (2013)	LTD13	3L-CVRP	n: 19-255; m: n.a.	96	French cities
Wei et al. (2014)	WTL14	3L-VRPHF	n: 20-100; m: 37-212	36	Golden et al. (1984)
Bortfeldt et al. (2015)	BHMM15	3L-VRPCB	n: 21-150; m: 37-461	95	GJB89, TV96
Pace et al. (2015)	PTMA15	3L-VRPHFTW	n: 130, 158	2	industry partner
Männel and Bortfeldt (2016)	MB16	3L-PDP	n: 104-212; m: 104-318	54	-
Reil et al. (2017)	RBM17	various 3L-VRPBTWs	n: 25-1,000; m: 1,025-50,000	192	MO09, BH13

3L-VRPHF(TW): 3L-VRP with a heterogeneous fleet (and time windows), abbrev.: abbreviation, basis: instances from the literature that serve as basis for the newly generated instances, n: number of customers per instance, m: total number of items, PPVRP: pallet-packing VRP with time windows, TSPLIB: VRP instances available in the TSPLIB (cf. Reinelt, 1991), #: number of instances

Chapter 4

Hybrid solution approach

The 3L-CVRP is a generalization of the VRP and as such an NP-hard combinatorial optimization problem (e.g. Lenstra and Kan, 1981). In addition, the packing problem included in the 3L-CVRP is a generalization of the bin packing problem, which is also NP-hard (e.g. Martello et al., 2000). Combining a VRP and a packing problem is, thus, a challenging optimization problem and difficult to solve (e.g. Iori and Martello, 2010). Being generalizations of the 3L-CVRP, the 3L-VRPBTWs regarded in this thesis are also difficult NP-hard problem. Thus, heuristic approaches are needed in order to obtain good solutions for instances of practically relevant size within reasonable computing times. More precisely, hybrid solution approaches are required for solving the integrated routing and packing problems. Such approaches for the 3L-VRPBTWs are introduced in the following sections.

4.1 Packing heuristics

If the routing subproblem of a 3L-VRPBTW is solved by means of a metaheuristic, several thousand iterations are typically performed during the search process and a vast number of routes must be tested for feasibility with respect to the packing constraints. An efficient and effective packing algorithm is needed, i.e. one that is able to find feasible OPP solutions for a huge variety of routes within very short computing times. Therefore, comparatively simple packing heuristics are employed for the presented hybrid algorithm.

In Chapter 4.1.1, definitions are given, which are required for the following considerations. Subsequently, two kinds of algorithms are presented. The first group of packing algorithms consists of simple and fast construction heuristics. The second group contains more complex algorithms that require longer computing times but are also able to find solutions to the OPP for routes where the simple heuristics fail. The different algorithms are presented in Chapter 4.1.2 and Chapter 4.1.3.

The heuristics have been implemented for solving the OPP. They can also be applied for solving a CLP. The required adjustments are motivated and outlined in Chapter 4.1.4.

The presented packing heuristics are based on heuristics from the literature. They have been extended substantially in order to cope with different loading approaches and consider various packing constraints. Depending on the applied loading approach (cf. Chapter 2.1.2), the packing plans can have different properties. All of the presented packing heuristics can be utilized for any loading approach. The necessary adaptations are described in Chapter 4.1.5. In Chapter 4.1.6, the implementation of each packing constraint is outlined in detail.

4.1.1 Definitions

In the following, the placement Π'_{ik} of an item I_{ik} is defined as a 4-tuple consisting of the placement coordinates and the chosen orientation:

$$\Pi'_{ik} = (\alpha_{ik}, \beta_{ik}, \gamma_{ik}, o_{ik}). \tag{4.1}$$

It constitutes a shortened form of the placement definition introduced in Chapter 2.1.1 (cf. (2.2)). Instead of the front-top-right corner point $(\alpha^0_{ik}, \beta^o_{ik}, \gamma^0_{ik})$, the orientation o_{ik} of an item is taken into account. It has the value 1 if the length l_{ik} of item I_{ik} is parallel to the α-axis, and 0 otherwise (cf. (2.25)). In the following algorithms, the shortened form can be used as the algorithms allow for neglecting *when* an item is loaded and unloaded during a route.

A *packing pattern* is used synonymously for an OPP solution generated by a packing heuristic. That is, a packing pattern is the output of solving an OPP for a given set of items. It contains a collection of Π'_{ik} for all items I_{ik} in the respective set, i.e. information about the positions and chosen spatial orientations of these items. In contrast to a packing plan, a packing pattern does not include information regarding the stages of a route where an item is loaded or unloaded (see Chapter 2.1.1).

Furthermore, the expression "a route (or an item sequence) can be *packed feasibly*" is used in order to state that a feasible packing plan for all included items of the route (or item sequence) exists.

For the sake of brevity, only one index is used (mostly) for the items in this chapter.

4.1.2 Construction heuristics

In the following, the employed construction heuristics are described. For the sake of simplicity, some figures depict 2D illustrations. In this case, the illustrated dimen-

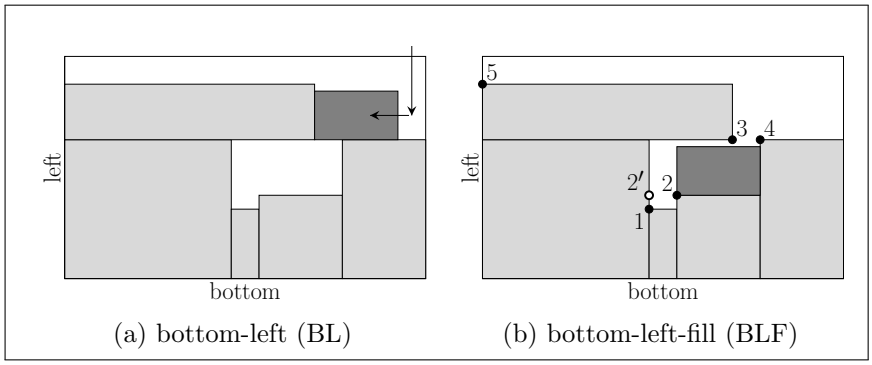
sions of the loading space refer to its width and height. The described heuristics can be applied to both 2D and 3D problems.

4.1.2.1 Deepest-bottom-left-fill heuristics

Firstly, different variants of the deepest-bottom-left-fill (DBLF) heuristic are presented. They are based on the bottom-left (BL) heuristic (Baker et al., 1980) and the bottom-left-fill (BLF) heuristic (Hopper, 2000), which were initially developed for the two-dimensional CLP, and the DBLF heuristic proposed by Karabulut and İnceoğlu (2005) for the three-dimensional CLP.

As the name suggests, the general idea of a BL heuristic is to place the items as far as possible to the bottom (first priority) and to the left (second priority). Applied to a three-dimensional packing problem, the items are to be placed as far as possible to the back (first priority), to the bottom (second priority) and to the left (third priority) of the loading space.

Whereas early implementations of the BL heuristic make use of sliding techniques, Hopper (2000) applies a different approach that allows the filling of gaps. Therefore, "*fill*" is appended to the names of such heuristics in order to distinguish them from the sliding approaches. Instead of sliding items, potential placement positions are used and the bottom- and left-most position where an item can be placed feasibly is chosen. Figure 4.1 illustrates the difference between the BL and the BLF approach.



(a) bottom-left (BL) (b) bottom-left-fill (BLF)

Figure 4.1: Comparison between BL and BLF approach (Adapted from Kopp, 2015)

The sliding technique is shown in Figure 4.1a as it is used, e.g., for the BL heuristic by Jakobs (1996) or Liu and Teng (1999). In Figure 4.1b, the BLF approach as implemented by Hopper (2000) is illustrated. Position 2 cannot be reached by sliding the item. Hence, gaps within the item arrangement can occur with the BL

approach, which cannot be filled up any more.

Karabulut and İnceoğlu (2005) have introduced the DBLF heuristic as an extension of the approach of Hopper (2000) for the three-dimensional CLP. The general procedure of the DBLF heuristic is outlined in Algorithm 4.1.[16]

Algorithm 4.1 DBLF heuristic for the OPP

Input: sorted sequence of items IS, current route R, instance data
Output: packing pattern, *result*

```
 1: procedure DBLF
 2:   result := TRUE                          ▷ TRUE: feasible packing pattern was found
 3:   initialize sorted set of positions P := {(0,0,0)}   ▷ sorted based on DBL-rule
 4:   K := ∅                                  ▷ set of already packed items and their placements
 5:   for iter := 1 to |IS| do                ▷ iter: iterator for IS
 6:     current item I_i := IS(iter)
 7:     placed := FALSE                       ▷ TRUE: feas. placement for the item was found
 8:     for p := 1 to |P| do
 9:       for each permitted orientation o do
10:         PLACEMENTTEST(I_i, P(p), o, K, R)
11:         if placement is feasible then
12:           placed := TRUE
13:           goto PLACEMENT
14:         end if
15:       end for
16:     end for
17:     PLACEMENT:
18:     if placed = TRUE then
19:       Π'_i := (P(p), o)                   ▷ store position and orientation for I_i
20:       SHIFTITEM(I_i, Π'_i, K, R)          ▷ Algorithm 4.2
21:       K := K ∪ {(I_i, Π'_i)}
22:       update P
23:     else
24:       result := FALSE
25:       break
26:     end if
27:   end for
28: end procedure
```

Initially, the items included in a packing pattern are sorted according to a pre-defined sorting rule. Subsequently, they are placed successively based on the resulting item sequence, hereinafter referred to as IS. The applied sorting rule is the following: First and foremost, the packing sequence depends on the sequence in which the customers are visited in the route. In order to facilitate the fulfilment of the LIFO-constraint, the linehaul items of the customer that is visited *last* are packed first

[16] In the following codes (i.e. after Algorithm 4.1), the instance data are assumed to be provided and, for the sake of brevity, not stated explicitly in the input data.

(see Chapter 2). Backhaul items that are picked up first are loaded first. Note that the presented packing procedures allow for generating separate packing patterns for linehaul and backhaul items (see below). Therefore, the sorting is conducted *either* in reverse delivery order *or* in pickup order. The items of one customer are then sorted by the following criteria:

- fragility (non-fragile items first), then
- volume (items with larger volume first), then
- length (longer items first), then
- width (wider items first).

Let P be the set of potential placement positions where an item can be placed with its bottom-left-back corner point. Initially, P contains only the origin of coordinates, i.e. the bottom-left-back corner of the loading space $(0, 0, 0)$, where the first item of the sequence is placed. After each placement, P is updated (Algorithm 4.1, line 22). That is, unusable points are deleted (those that are covered by the placed item) and new positions are added. The new positions are the top-left-back, bottom-left-front and bottom-right-back corner of the placed item. They are only added if a placement is basically possible, i.e. if at least the smallest item can be placed there. Let l_{min} and h_{min} be the shortest length/width and height, respectively, of any item I_{ik} $(i \in N_c, k \in J_i)$:

$$l_{min} = \min_{i \in N_c, k \in J_i} \min\left(l_{ik}, w_{ik}\right) \tag{4.2}$$

$$h_{min} = \min_{i \in N_c, k \in J_i} h_{ik} \tag{4.3}$$

Due to the permitted rotations, only the two measures l_{min} and h_{min} are necessary. A point (α, β, γ) can only be included in P if $L - \alpha \geq l_{min}$, $W - \beta \geq l_{min}$ and $H - \gamma \geq h_{min}$. The positions in P are sorted lexicographically according to the DBL-priority rule.

Starting with the first position in P, the positions are successively tested for feasibility when an item is to be placed. These tests ensure a feasible placement of the items with respect to the packing constraints presented in Chapter 2 and will be referred to as *placement tests* in the following. A placement test contains feasibility checks for all considered packing constraints. The individual tests are presented in detail below (Chapter 4.1.6).

If necessary, both permitted rotations of an item are tested (line 9). First, the orientation $o_i = 1$ of the current item I_i is tested, i.e. the item is rotated so that its edge with the measure l_i (provided in the instance data) is parallel to the α-axis. Subsequently, $o_i = 0$ is tested (if necessary), i.e. the item is rotated so that its edge with the measure l_i is parallel to the β-axis. Considering $o_i = 0$ can be omitted if $l_i = w_i$. As soon as a feasible placement (position and orientation) is found for the current item, no further positions and orientations are tested. The first item in the sequence IS is always placed at position $(0, 0, 0)$.

A position where no further movement towards the back, bottom and left is possible is called DBL-stable. Being placed in a position from P might not necessarily result in a DBL-stable placement. This is caused by the definition of the points added to P in every iteration. In order to ensure that an item is indeed placed as far as possible to the back, bottom and left, it is further shifted towards such a position (if possible) after a successful placement test, i.e. after a feasible (starting) placement position was identified (see Algorithm 4.1, line 20). This is done by applying the procedure outlined in Algorithm 4.2.

Algorithm 4.2 Shifting an item to a final placement as part of the DBLF heuristic

Input: current item I_i with placement Π'_i, set of already packed items and their placements K, current route R

Output: final placement Π'_i

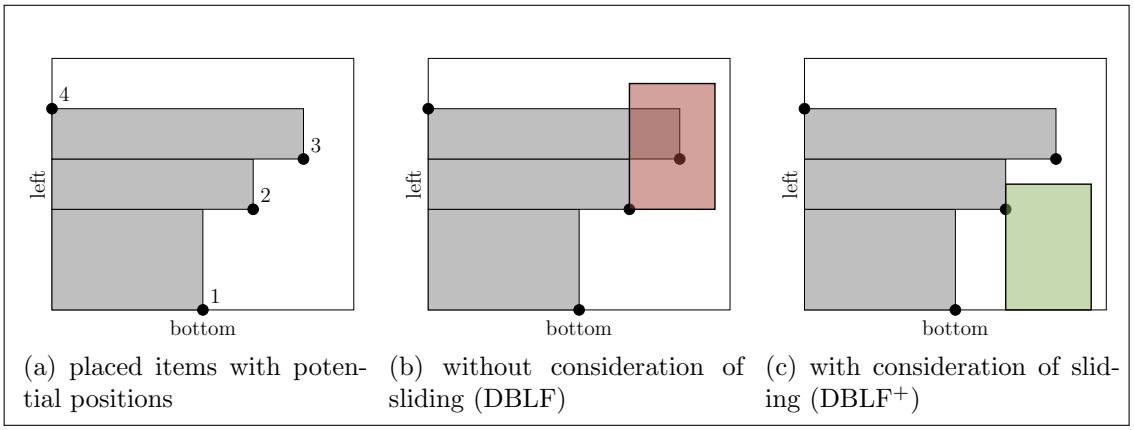
```
1: procedure ShiftItem
2:   while further movements are possible do
3:     if movement towards back is possible then
4:       move item as far as possible to the back
5:       update α_i ∈ Π'_i if necessary
6:     else if movement towards bottom is possible then
7:       move item as far as possible to the bottom
8:       update γ_i ∈ Π'_i if necessary
9:     else if movement towards left is possible then
10:      move item as far as possible to the left
11:      update β_i ∈ Π'_i if necessary
12:    end if
13:  end while
14: end procedure
```

Hence, the final placement position of an item is not necessarily the starting position from P. If necessary, multiple iterations of the shifting procedure are conducted until a final position is reached. Further movements can either be restricted by the faces of other items or by the loading space walls, or if the movement would lead to a

position that violates another packing constraint. For example, an item cannot be moved behind an item that is delivered later. For example, position $2'$ in Figure 4.1b is not included in P but is reached by placing the item in position 2 and then shifting it further towards the left.

The DBLF heuristic terminates as soon as all items are placed or no feasible placement can be found for an item.

Different variants of implementing the DBLF heuristic will be tested. Apart from the approach described above as it was implemented by Karabulut and İnceoğlu (2005), an extension of the approach is also regarded, which was introduced by Kopp (2015). It will be referred to as DBLF$^+$ in the following. In this extension, sliding the item towards a (D)BL-stable position is already considered during the placement test. Hereinafter, the corresponding procedure will be referred to as PlacementTest$^+$. That way, the items could be placed in feasible positions that would not be obtained by the original approach. An example is illustrated in Figure 4.2.



(a) placed items with potential positions (b) without consideration of sliding (DBLF) (c) with consideration of sliding (DBLF$^+$)

Figure 4.2: Illustration of DBLF$^+$ (Adapted from Kopp, 2015)

Position 1 (see Figure 4.2a) would be the first to be tested for the placement of the current item, because it is the bottom-most position. This placement would be infeasible, though, since the item would overlap with other items. Thus, position 2 is tested next. Using the approach of Hopper (2000), this position would lead to an infeasible placement as well (Figure 4.2b). However, as mentioned above, sliding an item towards a BL-stable position is already considered during the placement test in the extension. Sliding the item further to the bottom leads to a feasible placement here (Figure 4.2c). In contrast to the movement that is conducted after a feasible position is found (cf. Algorithm 4.1, line 20), the item is slid into only *one* direction during the placement test of DBLF$^+$. In doing so, sliding towards the back is tried

first, then towards the bottom and then towards the left.

In addition, a combination of both approaches is also considered (DBLF-Comb) in which the extension DBLF$^+$ is only used if the procedure with the original placement test cannot find a feasible placement for an item (Kopp, 2015). In that case, all positions are tested all over again with PlacementTest$^+$ (Algorithm 4.3).

Algorithm 4.3 DBLF-Comb heuristic for the OPP

Input: sorted sequence of items IS, current route R
Output: packing pattern, *result*

```
 1: procedure DBLF-Comb
 2:   result := TRUE                          ▷ TRUE: feasible packing pattern was found
 3:   initialize sorted set of positions P := {(0, 0, 0)}
 4:   K := ∅                                  ▷ set of already packed items and their placements
 5:   for iter := 1 to |IS| do                ▷ iter: iterator for IS
 6:     current item I_i := IS(iter)
 7:     placed := FALSE                       ▷ TRUE: feas. placement for the item was found
 8:     for p := 1 to |P| do
 9:       for each permitted orientation o do
10:         if PlacementTest(I_i, P(p), o, K, R) is feasible then
11:           placed := TRUE
12:           goto Placement
13:         end if
14:       end for
15:     end for
16:     if placed = FALSE then
17:       for p := 1 to |P| do
18:         for each permitted orientation o do
19:           if PlacementTest+(I_i, P(p), o, K, R) is feasible then
20:             placed := TRUE
21:             goto Placement
22:           end if
23:         end for
24:       end for
25:     end if
26: Placement:
27:     if placed = TRUE then
28:       Π'_i := (P(p), o)▷ store position (with sliding if necessary) and orientation
29:       ShiftItem(I_i, Π'_i, K, R)
30:       K := K ∪ {(I_i, Π'_i)}
31:       update P
32:     else
33:       result := FALSE
34:       break
35:     end if
36:   end for
37: end procedure
```

4.1.2.2 Touching area heuristics

The second group of construction heuristics are touching area (TA) heuristics. They are based on the touching perimeter algorithm that was first presented by Lodi et al. (1999) for a 2D packing problem. The first item is placed in the most bottom-left corner. All further items are placed in positions that maximize the percentage of the item perimeter that touches other, previously placed items or the container walls. Analogously, in the 3D case, the touching area of the item's surfaces is considered (cf. e.g. Gendreau et al., 2006). The touching areas of an item in a certain position are exemplarily illustrated in Figure 4.3. The blue (1) and green areas (2) are touching areas with items, whereas the orange area (3) is touching the container wall. The sum of these areas describes the TA value for a certain item in a certain position. An item is placed in the position with the maximum TA value. As in the DBLF procedure, the items are pre-sorted based on the above mentioned sorting rule (p. 110) and placed according to the resulting sequence. The procedure is outlined in Algorithm 4.4.

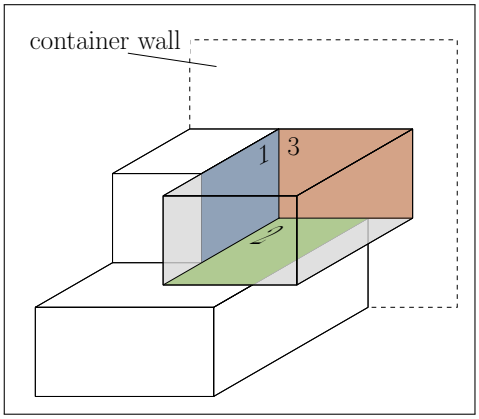


Figure 4.3: Example for the determination of touching areas

In contrast to the DBLF-algorithm, searching for a position does not stop as soon as one feasible position is found, but *all* potential positions and possible orientations are evaluated. Therefore, it usually entails longer computing times.

Different variants of the TA heuristic are regarded. The first variant, called TA-Walls, takes both the touching areas with other items as well as the touching areas with the container walls into account. The second variant – TA-NoWalls – considers only the touching areas with other items. With respect to the placement tests, the same extension as for the DBLF approach can be applied (cf. Figure 4.2). That is, PlacementTest$^+$ is called in the course of the TA heuristic (line 10). The

respective heuristics are referred to as TA$^+$-Walls and TA$^+$-NoWalls. Unlike before, a combination of both approaches is not considered.

Algorithm 4.4 TA heuristic for the OPP

Input: sorted sequence of items IS, current route R
Output: packing pattern, *result*

```
 1: procedure TouchingArea
 2:   result := TRUE                          ▷ TRUE: feasible packing pattern was found
 3:   initialize sorted set of positions P := {(0,0,0)}
 4:   K := ∅                                  ▷ set of already packed items and their placements
 5:   for iter := 1 to |IS| do                ▷ iter: iterator for IS
 6:     current item I_i := IS(iter)
 7:     ta' := −1                             ▷ maximum TA value
 8:     for p := 1 to |P| do
 9:       for each permitted orientation o do
10:         PlacementTest(I_i, P(p), o, K, R)   ▷ or PlacementTest+
11:         if placement is feasible then
12:           determine touching area value ta
13:           if ta > ta' then
14:             ta' := ta, p' := p, o' := o
15:           end if
16:         end if
17:       end for
18:     end for
19:     if ta' > −1 then                      ▷ a feasible placement was found
20:       Π'_i := (P(p'), o')                 ▷ store position and orientation for I_i
21:       ShiftItem(I_i, Π'_i, K, R)
22:       K := K ∪ {(I_i, Π'_i)}
23:       update P
24:     else
25:       result := FALSE
26:       break
27:     end if
28:   end for
29: end procedure
```

4.1.2.3 Open space heuristic

The open space (OS) heuristic is presented by Zhang et al. (2015). Initially, the set of items to be packed is sorted. Two sorting rules are adapted from Wei et al. (2014). They are similar to the sorting rule described in Chapter 4.1.2.1. In the first variant, the items are sorted primarily according to the customer sequence in the route (if the LIFO constraint is applied). Secondly, they are sorted by fragility; then, by non-increasing base area, and finally by non-increasing volume. The second sorting rule is similar. The only difference is that the length of the items is considered as

the third sorting criterion instead of the base area.

The packing heuristic works with open spaces representing possible positions for the items. These are cuboid spaces where one face lies at the rear side. An open space is described by its coordinates (α, β, γ) of the closest corner to the origin and its length (l), width (w) and height (h). During the search procedure, the current open spaces are sorted by ascending α-coordinate, then by ascending β-coordinate, then by ascending γ-coordinate. Thus, it can be seen as a deepest-left-bottom packing procedure.

The set *spaceList* is updated after every placement. That is, the used space *sp* is removed; up to three sub-spaces of *sp* are included, and the spaces that intersect with the placed item and those that are dominated by other spaces are removed. A space a is said to be dominated by another space b if they have the same coordinates and if a is totally contained by b, i.e. the edges (length, width, height) of a are not longer than the respective edges of b.

Algorithm 4.5 Open space packing heuristic (Adopted from Zhang et al., 2015)

Input: sorted sequence of items IS, current route R
Output: packing pattern K, number of packed items $|K|$

procedure OpenSpaceHeuristic
  $sp.\alpha = 0, sp.\beta = 0, sp.\gamma = 0, sp.\text{w} = \text{W}, sp.\text{l} = \text{L}, sp.\text{h} = \text{H}$
  $spaceList := \{sp\}$
  $K := \emptyset$ ▷ set of placed items
  while $spaceList \neq \emptyset$ and $IS \neq \emptyset$ **do**
    I_i = first non-packed item in IS
    find the first space *sp* that can accommodate I_i
    if such space *sp* is found **then**
      place I_i at *sp* and remove *sp* from *spaceList*
      $spaceList := spaceList \setminus \{sp\}$
      $IS := IS \setminus \{I_i\}$
      $K := K \cup \{(I_i, \Pi_i')\}$
      update *spaceList*
    else
      break
    end if
  end while
end procedure

In the course of this thesis, the OS heuristic is not used as a separate construction heuristic. It is only regarded in connection with the local search (LS) procedure presented below.

4.1.3 Local search framework

In addition to the construction heuristics, a more complex LS-based approach is tested as well. The procedure is presented by Zhang et al. (2015). In the course of this thesis, it is extended by integrating different construction heuristics, which can be selected alternatively. For this purpose, the construction heuristics are adapted in the way that they return the number of packed items (like the open space heuristic from Chapter 4.1.2.3).

The packing heuristic consists of two components: A local search is employed in order to generate different item sequences, and a packing construction heuristic is used to check whether a certain sequence of items can be packed feasibly. The whole procedure, called *LocalSearchPack*, is outlined in Algorithm 4.6.

Two cases are distinguished based on the number of items $|IS|$ in one packing pattern of a route R. In the first case (lines 4 to 9), where the number of items is not larger than a pre-defined parameter max_enum (Zhang et al., 2015, suggest a value of 8), a packing construction heuristic (here: a variant of DBLF, TA or OS) is called on every permutation of the items. The construction heuristic is aborted as soon as one item cannot be placed feasibly or all items are packed. It returns the number of items p that could be packed feasibly. If p equals the number of items of the route $|IS|$, feasible placements are found for all items of the route.

In the second case (lines 11 to 33), only a part of the permutations of the item sequence IS is tested by means of a local search procedure. Initially, a sorting rule is used for generating a sequence of items IS. Multiple sorting rules may be used in the course of the search, e.g. Zhang et al. (2015) use the two rules presented in Chapter 4.1.2.3. Then, a packing construction heuristic is called on IS and returns the number of items p that could be packed. Again, if p equals $|IS|$ the procedure stops with a feasible packing pattern. Otherwise, it tries to increase p by applying a random local search to the sequence of items IS. In that case, a new sequence IS^* is generated by randomly swapping two items in the sequence. This new sequence is accepted, if it is not worse than the previous one, i.e. if the packing heuristic returns at least as many packed items as before (lines 21 to 23). If the packing heuristic returns $|IS|$ packed items for any sequence, LocalSearchPack can be stopped. A feasible packing pattern is found.

Originally, LocalSearchPack is presented by Zhang et al. (2015) with an inte-

grated OS heuristic. Pre-tests have shown that the procedure requires considerably more computing time than the construction heuristics described in Chapter 4.1.2. In the hope of obtaining another effective and fast LS-heuristic, the construction heuristics from Chapter 4.1.2 are implemented into the LS-based approach. They replace the open space heuristic (that is called in LocalSearchPack (lines 5 and 20)) and the sorting rule from Chapter 4.1.2.1 is applied (cf. p. 110).

Algorithm 4.6 LS packing heuristic (Adapted from Zhang et al., 2015)

Input: set of items to be packed IS, current route R, parameter max_enum
Output: packing pattern, $result$

```
1: procedure LocalSearchPack
2:   result := FALSE                          ▷ TRUE: feasible packing pattern was found
3:   if |IS| ≤ max_enum then
4:     for each permutation IS* of IS do
5:       if Packing(IS*, R) = |IS| then       ▷ selected packing construction
                                                heuristic returning the number of packed items
6:         result := TRUE                     ▷ feasible packing pattern was found
7:         break
8:       end if
9:     end for
10:  else
11:    for each sorting rule do
12:      sort IS according to the sorting rule
13:      p := Packing(IS, R)
14:      if p = |IS| then
15:        result := TRUE                     ▷ feasible packing pattern was found
16:        break
17:      else
18:        for k := 1 to |IS| do
19:          generate a new sequence IS* by swapping two randomly selected
             items in IS
20:          p* := Packing(IS*, R)
21:          if p* ≥ p then
22:            IS := IS*, p := p*
23:          end if
24:          if p = |IS| then
25:            result := TRUE                 ▷ feasible packing pattern was found
26:            break
27:          end if
28:        end for
29:      end if
30:      if result = TRUE then
31:        break
32:      end if
33:    end for
34:  end if
35: end procedure
```

In conclusion, the construction heuristics and LS-based heuristics as shown in Table 4.1 are utilized. The heuristics in combination with the LS are hereinafter referred to as LS_DBLF, LS_TA-Walls, LS_OS etc.

Table 4.1: Implemented packing heuristics

group	**heuristic**	**individually as construction heuristic**	**in combination with the LS (LS_*)**
DBLF	DBLF	✓	✓
	DBLF+	✓	✓
	DBLF-Comb	✓	✓
TA	TA-Walls	✓	✓
	TA-NoWalls	✓	✓
	TA^+-Walls	✓	✓
	TA^+-NoWalls	✓	✓
OS	OS		✓

4.1.4 Adaption to the CLP

As mentioned above, the packing heuristics are implemented in order to solve the OPP as part of the 3L-VRPBTW. In order to evaluate the suitability of the heuristics for a combined routing and packing procedure, the heuristics are also applied to CLP instances from the literature and compared to a state-of-the-art CLP heuristic. (see below, Chapter 5.5.1.3). In order to apply the heuristics to the CLP, the following modifications are necessary:

- The heuristics are not aborted as soon as one item cannot be placed. Instead, the item is skipped and not packed in the final solution.
- Depending on the problem definition, all six orientations can be considered. They can also be limited for certain items. Let l_0, w_0, h_0 be the original edge dimensions given in the instance data, and (l, w, h) be the edges used as length, width and height, respectively, according to the chosen orientation (i.e. the edges of placed items parallel to the length, width and height axes of the loading space). The rotations are tested in the following order: (l_0, w_0, h_0), (w_0, l_0, h_0), (l_0, h_0, w_0), (h_0, l_0, w_0), (h_0, w_0, l_0), (w_0, h_0, l_0).
- Within the LS framework, the packed volume instead of the number of packed items is maximized as the packed volume is the usual optimization criterion in the CLP.

4.1.5 Implementation of the loading approaches

Different loading approaches are applied to the different problem variants (see Chapter 2.1.2). In the following, aspects considered for their implementation are noted.
The basic loading approach underlying all packing programmes, is the rear loading approach. Thus, the heuristics need to be adjusted in order to realize the approaches loading space partition (LSP) and side loading.
In order to apply the rear loading approach to the 3L-VRPTW, only one packing pattern is required per route as only linehaul items are transported. This pattern represents the pattern at the beginning of the route. The patterns for the following stages, result from unloading items. If it is applied to the 3L-VRPCB(TW), two separate packing patterns are generated for the linehaul (for the beginning of the route) and backhaul customers (for the end of the route). The only difference in their generation is the adaptation of the sorting rules for the items. If a packing pattern is created for linehaul customers, the items are sorted in the reverse delivery order. In the case of backhaul customers, they are sorted in the order of the pickup.
The LSP is implemented similarly to the rear loading approach with the difference that the height of the loading space as an input datum for the packing programmes is halved. If an output of the results is desired, the γ-coordinates of the items of one type (linehaul or backhaul) must be increased by $H/2$. Two separate packing patterns are created for linehaul and backhaul items, which differ in the sorting rule of the items (see above).
The implementation of the side loading approach depends on the problem variant. If linehaul and backhaul items are transported separately (3L-VRPTW/3L-VRPCB(TW)), the approach is realized by swapping the length and width dimensions of the loading space. Consequently, the packing problem can be solved like a problem with rear loading with separate packing patterns for linehaul and backhaul items if necessary (3L-VRPCB(TW)).
If linehaul and backhaul items are transported simultaneously (3L-VRPMB(TW)/3L-VRPSDP(TW)), the side loading approach is implemented in a way that linehaul and backhaul items are loaded at the opposing sides of the loading space (rear and front). By unloading the linehaul items in the course of the route, space is created sequentially for the backhaul items. For this purpose, two separate packing patterns are created for the linehaul and backhaul items. Both patterns must be feasible and

together they define the packing plan for the corresponding route. In order for the whole packing plan to be feasible, the linehaul and backhaul items must not overlap at any stage of a route.

The side loading approach is explained using an example instance for the 3L-VRPMBTW. An exemplary route is illustrated in Figure 4.4.

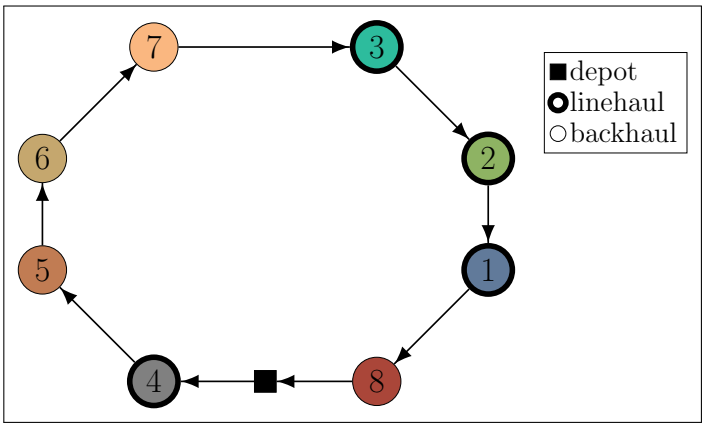


Figure 4.4: Example 3L-VRPMBTW route

As before, the generation of the packing patterns is identical for linehaul and backhaul items apart from the applied sorting rule. Due to the modus operandi of the packing heuristics, the items of both types are initially packed at the back of the loading space ($W \times H$ plane at $(0, 0, 0)$). Figure 4.5 shows the packing patterns that are generated for the linehaul (Figure 4.5a) and the backhaul items (Figure 4.5b) in the example route.

In order to obtain a packing plan where the items are loaded at opposing sides, the placements of one group of items must be mirrored. Hereinafter, it is assumed that the linehaul items are loaded at the driver's cabin ($W \times H$ plane at $(0, 0, 0)$) and the backhaul items are loaded at the rear side ($W \times H$ plane at $(L, 0, 0)$). Let $(\alpha'_i, \beta'_i, \gamma'_i)$ be the position of the bottom-left-back corner of an item I_i that was determined by the applied packing heuristic. If this item is a backhaul item, its placement in the final packing plan is at position $(\alpha_i, \beta_i, \gamma_i)$ with:

$$\alpha_i = L - \alpha'_i - \widehat{l}_i, \tag{4.4}$$

$$\beta_i = \beta'_i, \tag{4.5}$$

$$\gamma_i = \gamma'_i. \tag{4.6}$$

As introduced above, $\widehat{l}_i$ refers to the edge of the item that is parallel to the α-axis

of the loading space ($\widehat{l_i} = \alpha_i + o_i \cdot l_i + (1 - o_i) \cdot w_i$). If the item is a linehaul item, its position $(\alpha_i, \beta_i, \gamma_i)$ equals $(\alpha'_i, \beta'_i, \gamma'_i)$.

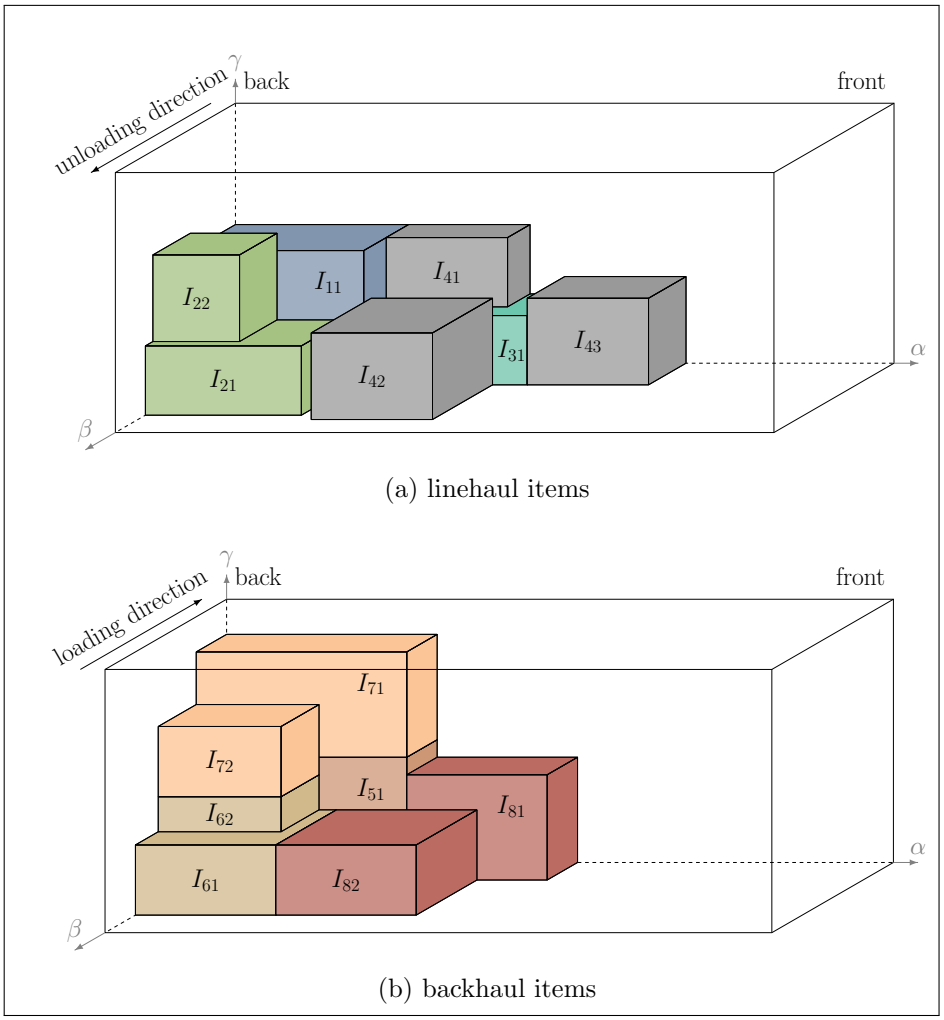


Figure 4.5: Separate packing patterns for linehaul and backhaul items

Figure 4.6 shows the items from the presented example that are in the loading space after customer 6 is visited. That is, the items of the linehaul customers 1, 2 and 3 are still loaded. In addition, the backhaul items of the customers 5 and 6 are already picked up. The backhaul items are loaded at the front side. In order to guarantee that the linehaul and backhaul patterns do not overlap at any stage of the route, the loading lengths of both sets (L_{LH} and L_{BH}) are determined.

Considering the packing patterns as they are provided by a packing heuristic, i.e. as they are depicted in Figure 4.5, the loading lengths refer to the maximum front faces of the respective linehaul or backhaul items. Transferred to the packing patterns with the converted positions (Figure 4.6), the loading length of the linehaul items is also determined as the maximum front face of the linehaul items. The loading length of the backhaul items refers to the loading space length minus the minimum

back face of the backhaul items in the loading space. The sum of both lengths must not exceed the loading space length L at any stage of the route.

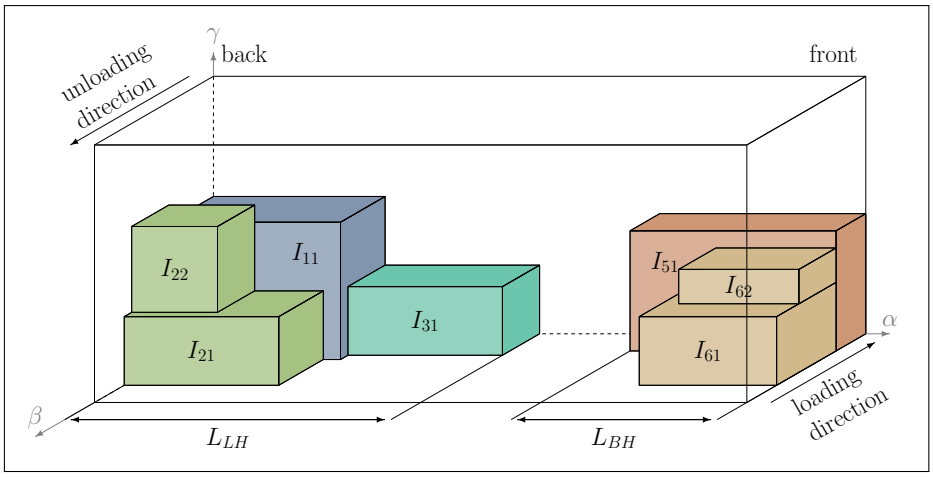


Figure 4.6: Item arrangement in a side loaded vehicle at a given stage of a route

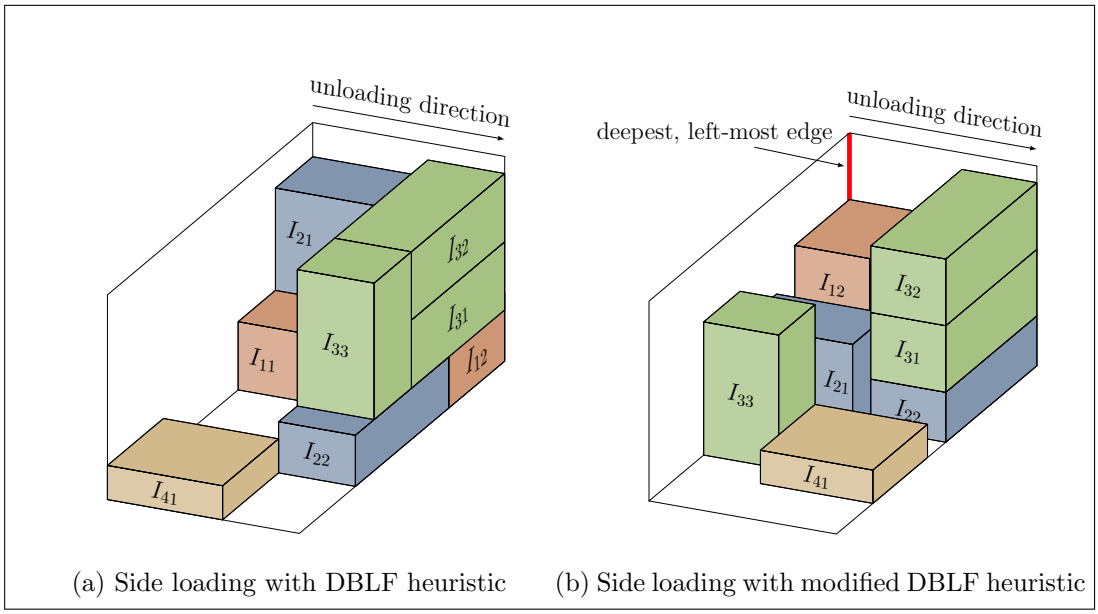
The LIFO constraint demands that an item that is delivered later (picked up earlier) must not be placed above items that are delivered earlier (picked up later) in a route, or between such items and the (un-)loading side. Applied to the side loading approach, the LIFO constraint must be considered along the width axis of the loading space, i.e. in (un-)loading direction. In the example in Figure 4.6, the items I_{21} and I_{22} must not be delivered after item I_{11} as they would block its unloading. In contrast to the application to the 3L-VRPTW and 3L-VRPCB(TW), the LIFO constraint is modified: it is also considered along the length axis in the presented implementation. That way, vertical layers are built and gradually creating space for backhaul items is facilitated. For example, a linehaul item delivered after item I_{31} (Figure 4.6) must not be placed at a greater α-coordinate.

For this purpose, the DBLF-based construction heuristics are adapted. By adjusting the LIFO constraint as explained above (considering it not only towards the loading side, but also parallel to it), the original DBLF approach results in packing patterns as depicted in Figure 4.7a. In this example, the customers are visited in the sequence $(4, 3, 2, 1)$. Due to considering the LIFO constraint from three sides of an item (top, towards loading side, parallel to loading side), gaps occur frequently. In Figure 4.7a, no item of the customers 3 and 4 must be placed *behind* item I_{22} (from the view of the loading side) resulting in a gap where no further item can be placed. That is, the loading space is not utilized efficiently. In addition, if horizontal stability was considered, the loading pattern would be rather unstable. Therefore, the

construction heuristics are modified regarding the sorting of the potential placement positions. Instead of sorting them according to the DBL rule, the first priority is to sort them by non-increasing distance from the deepest, left-most edge of the loading space. The distance of a point (α, β, γ) to the deepest, left-most edge is defined as the Manhattan distance, i.e.:

$$distance(\alpha, \beta, \gamma) = \alpha + \beta. \tag{4.7}$$

Ties are broken according to the DBL rule again. This modification results in packing patterns where the items of a customer tend to be stacked before they are placed next to each other. The items from the previous example would be arranged as illustrated in Figure 4.7b. In the following, this variant of the DBLF heuristic will be referred to as DBLF$^{\text{SL}}$ (and, accordingly, DBLF$^{+,\text{SL}}$ and DBLF$^{\text{SL}}$-Comb).



(a) Side loading with DBLF heuristic (b) Side loading with modified DBLF heuristic

Figure 4.7: Side loading with different implementations of the DBLF heuristic

4.1.6 Implementation of the packing constraints

In the following, aspects of the implementation of the packing constraints presented in Chapter 2 are explained. The focus will be on the implementation of the packing constraints *robust stability*, *load bearing strength* and *reachability* as they have not been studied as frequently as the other constraints in the literature. All of the following remarks refer to the generation of a packing pattern in the course of solving an OPP.

Not only the back-left-bottom corner point $(\alpha_i, \beta_i, \gamma_i)$ of an item I_i is required for the implementation, but also the opposite point $(\alpha_i^o, \beta_i^o, \gamma_i^o)$, i.e. the point at the front-right-top corner of item I_i (see Figure 2.3). Using the notation introduced above in (2.26)-(2.27), its coordinates can be determined as:

$$\alpha_i^o = \alpha_i + \widehat{l}_i = \alpha_i + o_i \cdot l_i + (1 - o_i) \cdot w_i, \tag{4.8}$$

$$\beta_i^o = \beta_i + \widehat{w}_i = \beta_i + (1 - o_i) \cdot l_i + o_i \cdot w_i, \tag{4.9}$$

$$\gamma_i^o = \gamma_i + h_i. \tag{4.10}$$

Geometrical constraints (P1)-(P3)

Compliance with the geometrical constraints is ensured by comparing the item coordinates and the loading space dimensions given in the instance data. An item is placed completely inside the loading space if:

$$(\alpha_i \geq 0) \wedge (\beta_i \geq 0) \wedge (\gamma_i \geq 0) \wedge (\alpha_i^o \leq L) \wedge (\beta_i^o \leq W) \wedge (\gamma_i^o \leq H)\,. \tag{4.11}$$

Furthermore, two items I_i and I_j do not overlap if:

$$\left(\alpha_i \geq \alpha_j^o\right) \vee (\alpha_j \geq \alpha_i^o) \vee \left(\beta_i \geq \beta_j^o\right) \vee (\beta_j \geq \beta_i^o) \vee \left(\gamma_i \geq \gamma_j^o\right) \vee (\gamma_j \geq \gamma_i^o)\,. \tag{4.12}$$

When an item I_i is to be placed, the adherence to (4.12) must be tested between item I_i and all other items that have already been placed. The orthogonal packing of each item is ensured by the choice of permitted orientations during the execution of the packing heuristics.

Fixed vertical orientation (P4)

As can be seen, for example, in Algorithm 4.1, only the permitted item rotations are tested. In the case of the OPPs considered here[17], the height dimension is fixed. Alternatively, all six spatial orientation could be tested if there are no restrictions (e.g. if a packing problem with the respective property is to be solved).

Vertical stability (P5)

The vertical stability of an item is examined by analogy with the considerations

[17] The different constraint sets C1-C5 define different OPPs (see Chapter 2.1.1).

made in Chapter 2.2 (p. 44ff.). The procedure is depicted in Algorithm 4.7.[18] The supported base area a_i of item I_i is composed of the sum of the individual base areas $supp_{ij}$ that are supported by the items I_j placed directly beneath I_i. An item I_j is placed *directly* below I_i, i.e. its top face touches the bottom face of I_i, if

$$\gamma_i = \gamma_j^o, \tag{4.13}$$

and if the projections of the items on the α-β-plane overlap, i.e.

$$\left(\alpha_i < \alpha_j^o\right) \wedge \left(\alpha_j < \alpha_i^o\right) \wedge \left(\beta_i < \beta_j^o\right) \wedge \left(\beta_j < \beta_i^o\right). \tag{4.14}$$

The areas $supp_{ij}$ are determined as (cf. (2.63) or (2.64)):

$$supp_{ij} = \left(\min\left(\alpha_i^o, \alpha_j^o\right) - \max\left(\alpha_i, \alpha_j\right)\right) \cdot \left(\min\left(\beta_i^o, \beta_j^o\right) - \max\left(\beta_i, \beta_j\right)\right). \tag{4.15}$$

If a_i is equal to or greater than the required supporting area ($VSP \cdot l_i \cdot w_i$), the constraint is satisfied.

Algorithm 4.7 Testing for vertical stability

Input: current item I_i with placement Π_i', set of already packed items and their placements K

Output: result of the test $feas$

```
1:  procedure CheckVertStability
2:    a_i := 0                                          ▷ supported base area
3:    feas :=TRUE     ▷ TRUE: placement is feasible w.r.t. the stability constraint
4:    if γ_i > 0 then                ▷ no test is necessary if I_i is placed on the floor
5:      for each (I_j, Π'_j) ∈ K do
6:        if I_j is directly below I_i then
7:          a_i := a_i + supp_ij      ▷ supp_ij: base area of I_i that is supported by I_j
8:        end if
9:      end for
10:     if a_i < VSP · l_i · w_i then              ▷ VSP: vertical stability parameter
11:       feas := FALSE
12:     end if
13:   end if
14: end procedure
```

[18] Note that the pseudocodes presented here in Chapter 4.1.6 depict separate loops. Several checks are implemented in one joint loop, though. The presentation differs here, in order to present the various implementations separately.

Fragility (P6)

Algorithm 4.8 illustrates how the compliance with the fragility constraint is ensured. Placing an item I_i, it must be guaranteed that no fragile items are placed directly (cf. (4.13), (4.14)) below it if I_i is not fragile ($f_i = 0$).

Algorithm 4.8 Testing for compliance with the fragility constraint

Input: current item I_i with placement Π'_i, set of already packed items and their placements K
Output: result of the test $feas$

```
 1: procedure CheckFragility
 2:   feas :=TRUE      ▷ TRUE: placement is feasible w.r.t. the fragility constraint
 3:   for each (I_j, Π'_j) ∈ K do
 4:     if I_j is directly below I_i then                    ▷ see (4.13), (4.14)
 5:       if f_j = 1 and f_i = 0 then          ▷ fragile item below non-fragile item
 6:         feas := FALSE                                    ▷ constraint is violated
 7:         break                                            ▷ terminate test
 8:       end if
 9:     else if I_i is directly below I_j then
10:       if f_i = 1 and f_j = 0 then
11:         feas := FALSE
12:         break
13:       end if
14:     end if
15:   end for
16: end procedure
```

As the newly placed item can also be placed underneath a previously placed item, compliance with the fragility constraint must not only be tested for items below the reference item, but also with items placed above it (lines 9ff.). If the constraint is violated for any two items I_i and I_j, the test is terminated.

LIFO (P7)

The LIFO constraint is violated if a linehaul (backhaul) item is placed in front of or above another linehaul (backhaul) item that is delivered (picked up) earlier (later) in the route. The loading approaches applied to the problem variants with simultaneous transport of linehaul and backhaul items allow for generating two separate packing patterns (see Chapter 4.1.5). Hence, it must not be considered in the implementation, for example, that a backhaul item must not be placed in front of, behind, above or underneath a linehaul item that is delivered after the backhaul item is picked up. If loading space partition is applied, this constraint is automatically satisfied due to the separate sections for linehaul and backhaul items.

In the case of the side loading approach, this aspect of the LIFO constraints is satisfied if the loading length of the linehaul and backhaul items do not exceed L. Testing for compliance with the LIFO constraint is outlined in Algorithm 4.9. In order to determine whether an item I_j is below (directly or indirectly) another item I_i, (4.14) can be applied with the addition that

$$\gamma_i \geq \gamma_j^o \tag{4.16}$$

must be fulfilled. That is, the surfaces of the items do not need to touch. Analogously, an item I_j is behind another item I_i if

$$\left(\alpha_i \geq \alpha_j^o\right) \wedge \left(\beta_i < \beta_j^o\right) \wedge \left(\beta_j < \beta_i^o\right) \wedge \left(\gamma_i < \gamma_j^o\right) \wedge \left(\gamma_j < \gamma_i^o\right). \tag{4.17}$$

These relations refer to the rear loading and LSP approach. For the side loading approach, α and β must be swapped.

Algorithm 4.9 Testing for compliance with the LIFO constraint

Input: current item I_i, Π_i', current route, set of already packed items and placements K
Output: result of the test $feas$

```
1:  procedure CheckLifo
2:    feas :=TRUE          ▷ TRUE: placement is feasible w.r.t. the LIFO constraint
3:    for each (I_j, Π'_j) ∈ K do
4:      if I_j is below I_i or I_j is behind I_i then
5:        if I_j is delivered before I_i or picked up after I_i then
6:          feas := FALSE                              ▷ constraint is violated
7:          break                                      ▷ terminate test
8:        end if
9:      else if I_i is below I_j or I_i is behind I_j then
10:       if I_i is delivered before I_i or picked up after I_j then
11:         feas := FALSE
12:         break
13:       end if
14:     end if
15:   end for
16: end procedure
```

Reachability (P8)[19]

The following explanations refer to the rear loading and LSP approach. The reacha-

[19] The test algorithms of the following constraints have been developed together with Corinna Krebs (Krebs (2017)).

bility constraint can be realized analogously for the side loading approach by swapping α- and β-coordinates.

In order to check whether an item is reachable in a certain placement, i.e. if the distance between the item and an operator does not exceed λ length units, a placement space is defined. Only those items that are placed within this placement space must be considered for the reachability of the current item. An item I_j is located within the placement space of I_i if:

$$\left(\alpha_j^o > \alpha_i^o\right) \wedge \left(\beta_i^o > \beta_j\right) \wedge \left(\beta_i < \beta_j^o\right). \tag{4.18}$$

The operator position δ_i refers to the maximum α-coordinate of the items inside the placement space of I_i (cf. (2.73)). An example is illustrated in Figure 4.8. The reference item is I_1 (yellow). The items I_2, I_3 and I_6 (green) are located within the placement space. The closest possible position of an operator when (un-)loading item I_1 is, thus, the front edge of I_2 ($\delta_1 = \alpha_2^o$).

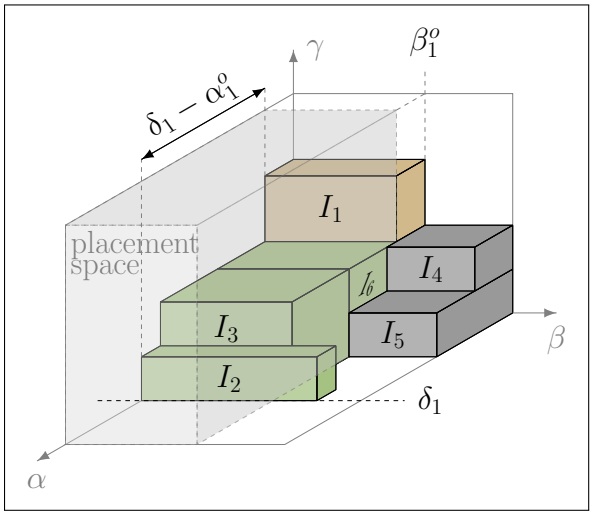


Figure 4.8: Illustration of the placement space for determining reachability

Moreover, only those items are relevant for the reachability of I_i that (in case of linehaul items) are either delivered at the same stage and placed below I_i, or that are delivered after I_i. In the first case, I_i would need to be unloaded before the lower items (see e.g. Figure 2.11). If a packing pattern is generated for backhaul items, items that are either picked up at the same stage and placed below I_i, or picked up earlier are relevant.

The framework for examining the feasibility of an item arrangement with respect to the reachability constraint is outlined in Algorithm 4.10. If the distance between

the placed item and the operator is too large (line 9), a placement is rejected.

Algorithm 4.10 Testing for reachability (rear loading)

Input: current item I_i with placement Π'_i, set of already packed items and their placements K
Output: result of the test $feas$

```
1:  procedure CheckReachability
2:    feas :=TRUE ▷ TRUE: placement is feasible w.r.t. the reachability constraint
3:    δ_i := α_i^o                          ▷ closest possible position of the operator
4:    for each (I_j, Π'_j) ∈ K do
5:      if I_j is within the placement space and I_j is relevant then
6:        δ_i := max(δ_i, α_j^o)
7:      end if
8:    end for
9:    if δ_i − α_i^o > λ then                          ▷ the distance is too large
10:     feas := FALSE
11:   end if
12: end procedure
```

In particular, if the differences between the item dimensions parallel to the α-axis are large, the reachability constraint can be restrictive. Therefore, an alternative approach is implemented. It is attempted to shift an item as far as possible (if necessary) towards the (un)loading side. This variant is outlined in Algorithm 4.11. The variable $MaxShift$ is employed in order to determine how far an item can be moved. The shift is only conducted in positive α-direction. That is, an item I_i can only be moved along the top faces of other items that are at the same level as the bottom face of I_i. $MaxShift$ takes the maximum α-value of the items I_j in the placement space for which $\gamma_i = \gamma_j^o$ is fulfilled (Algorithm 4.11, line 8). If the item is not reachable in the original position (line 12) but can be made reachable by shifting it (line 14), the shift is conducted and the new placement is tested for feasibility. The item is not necessarily shifted up to the coordinate $MaxShift$ but only as far as necessary (line 15). In doing so, the principle of placing an item as far as possible to the back is still observed and more space is left available for further items.

An example is depicted in Figure 4.9. Item I_1 cannot be reached unless it is shifted as the operator cannot step closer towards it than the front edge of I_2 (cf. Figure 4.8). Since items I_3 and I_6 have the same height, I_1 could be shifted until its front edge is at level with the front edge of I_3 ($MaxShift$). However, it is only shifted until the distance between its front edge and the operator position (δ_1) equals λ.

The implementation variant including the shift is applied in the experiments described in Chapter 5 since it proved superior (compared to the approach without

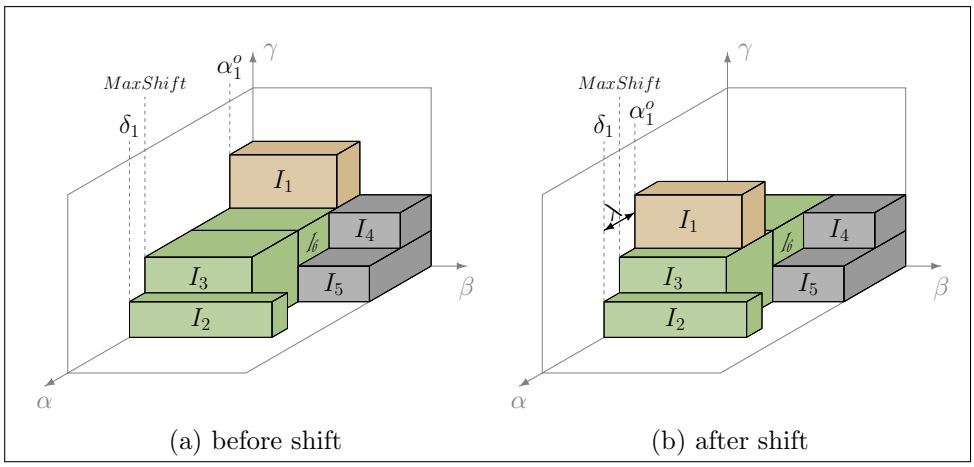
the shift) in pre-tests.

Algorithm 4.11 Testing for reachability with shifting (rear loading)

Input: current item I_i with placement Π'_i, set of already packed items and their placements K

Output: result of the test $feas$, Π'_i ▷ position of I_i might change

```
1: procedure CheckReachabilityShift
2:   feas :=TRUE   ▷ TRUE: placement is feas. w.r.t. the reachability constraint
3:   δ_i := MaxShift := α_i^o
4:   for each (I_j, Π'_j) ∈ K do
5:     if I_j is within the placement space and I_j is relevant then
6:       δ_i := max(δ_i, α_j^o)
7:       if γ_i = γ_j^o then
8:         MaxShift := max(MaxShift, α_j^o)
9:       end if
10:    end if
11:  end for
12:  if δ_i − α_i^o > λ then
13:    feas := FALSE
14:    if δ_i − MaxShift ≤ λ then
15:      α_i^o := δ_i − λ, α_i := α_i^o − l̂_i
16:      if shifted placement is feasible then
17:        feas := TRUE
18:      end if
19:    end if
20:  end if
21: end procedure
```



(a) before shift (b) after shift

Figure 4.9: Relevant coordinates for shifting an item to obtain reachability

Robust stability (P9)

The constraint demanding robust stability constitutes an extension of the vertical stability constraint. It has to be ensured that an item is sufficiently supported by

all items placed in a stack beneath the current item. The procedure for testing the compliance with the robust stability constraint is outlined in Algorithm 4.12.

Algorithm 4.12 Testing for robust stability

Input: current item I_i with placement Π'_i, set of already packed items and their placements K

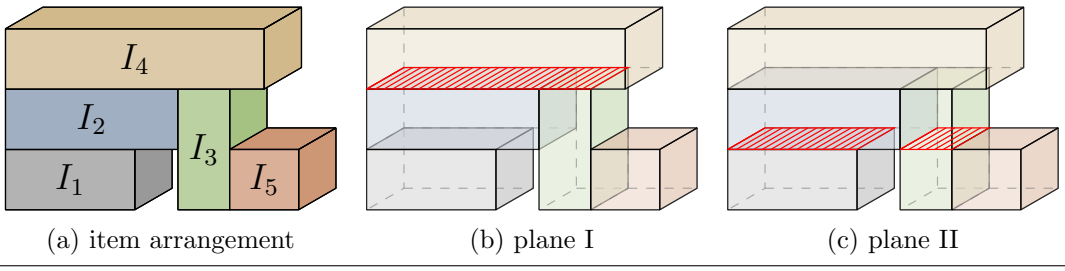
Output: result of the test $feas$

```
1:  procedure CheckRobustStability
2:    feas :=TRUE ▷ TRUE: placement is feasible w.r.t. robust stability constraint
3:    if γ_i > 0 then
4:      determine J_i^supp                          ▷ set of all items supporting I_i
5:      for each item I_j ∈ J_i^supp do
6:        a := supp_ij                              ▷ supported bottom area per plane
7:        for each item I_k ∈ J_i^supp \ {I_j} do
8:          if γ_k^o ≥ γ_j^o and γ_k < γ_j^o then   ▷ if I_k contributes to the plane of I_j
9:            a := a + supp_ik
10:         end if
11:       end for
12:       if a < VSP · l_i · w_i then
13:         feas := FALSE
14:         break
15:       end if
16:     end for
17:   end if
18: end procedure
```

First, the items (directly or indirectly) supporting the reference item I_i are determined. They are stored in the set J_i^{supp} (line 4). An item I_j supports I_i directly if it is placed directly below it (cf. (4.13), (4.14)). An item I_j supports I_i indirectly, if it directly supports any item that directly supports I_i. Moreover, if an item I_j directly supports any item that indirectly supports item I_i, I_j also supports I_i indirectly. Subsequently, the support a for each *plane* of the stack below item I_i is calculated (lines 6 and 9, cf. (4.15)). The planes are defined by the top faces of the items supporting the reference item. An item I_j contributes to the support in a given plane at γ_p if its top face is at the respective γ-coordinate of the plane ($\gamma_j^o = \gamma_p$) or if its top face is above the γ-coordinate of the plane and its bottom face is below it, i.e. $\gamma_j < \gamma_p < \gamma_j^o$. If a is lower than $VSP \cdot l_i \cdot w_i$ (i.e. the minimum area that must be supported) at any plane, the robust stability constraint is violated and the test is terminated. The supporting area $supp_{ij}$ provided by item I_j for I_j is determined as above (see (4.15)).

An example is given in Figure 4.10 where the reference item is I_4. One supporting plane is formed by the top faces of I_2 and I_3 at $\gamma_2^o = \gamma_3^o$ (Figure 4.10b). Another

plane is defined at γ_1^o, the top face of I_1. Further points on this plane and below I_4 are occupied by I_3 (Figure 4.10c). Thus, I_3 offers support for I_4 in this plane, too. Although I_5 is also located below I_4 and its top face is at the plane at γ_1^o, it does not support I_4 in any way. Therefore, it is not considered for the robust stability of I_4, i.e. it is not included in J_4^{supp}. The supporting areas per plane are marked red in Figures 4.10b and 4.10c.



(a) item arrangement (b) plane I (c) plane II

Figure 4.10: Determination of planes for robust stability

Load bearing strength (P10)

Placing an item adds load onto all items directly and indirectly supporting it. Therefore, the bearing loads (see below) per unit area of all of these items must be determined in order to check the feasibility of an item arrangement with respect to the load bearing strength constraint.

In the following, the *bearing load* represents the weight that an item has to carry from the item(s) placed directly and indirectly above it. The *load bearing strength* p_i of an item I_i describes the maximum admissible weight it can bear on any unit area of its top face. The following explanations are based on the variant of the load bearing strength constraint that considers the *static* load transmission (see Chapter 2.2.1, e.g. Figure 2.34, (2.80)). That is, it is assumed that the weight of an item is not only transmitted directly downwards, but towards all items supporting the reference item.

Let set J_j^{dsupp} contain all items that are directly supporting item I_j. That is, it contains all items I_k with $\gamma_j = \gamma_k^o$ and whose α-β-projections overlap with those of I_j (cf. (4.13), (4.14)).

In addition, set J_i^{LBS} contains all items that must be taken into consideration when placing an item I_i. For this purpose, the affected items (which can be more than the placed items and its supporting items) need to be identified. An example is illustrated in Figure 4.11. Here, item I_{10} (blue) is the current item. The relevant

items for testing the load bearing strength constraint are shaded in green. The weight of item I_{10} is transmitted directly to items I_8 and I_9, and indirectly to I_3, I_4 and I_6. Although, I_3 is not placed below I_{10} (i.e. with overlapping α-β-projection), the weight of I_{10} is transferred to it by I_6. Moreover, I_{10} does not put weight on item I_7. This item must be considered nonetheless. The weight of I_7 is also passed towards the items I_3, I_4 and I_6. Item I_7 must, thus, be taken into account when calculating the bearing load of those items. The remaining items (red) are not affected by the placement of item I_{10}. Hence, the load transmitted from and towards them does not have to be (re-)calculated. In conclusion, set J_i^{LBS} contains

- the reference item I_i (I_{10} in the example),
- items that support I_i directly and indirectly (I_3, I_4, I_6, I_8, I_9 in the example),
- items that are – directly or indirectly – supported by I_i,
- items that are supported by other items in J_i^{LBS} (I_7 in the example).

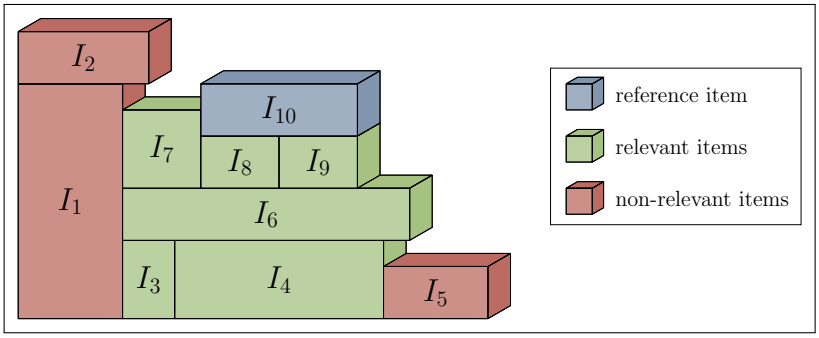


Figure 4.11: Relevant items for determining bearing loads based on a reference item

The general procedure is as follows: Let I_j be any item from J_i^{LBS}. First, the weight of I_j is distributed proportionally to each item I_k directly supporting I_j and added to the bearing load of the respective unit areas on the top face of I_k. Then, each item I_k supporting I_j transmits the load it bears (from I_j) towards each item directly supporting I_k and the bearing loads on their top faces are updated. This procedure is repeated until the weight of I_j is transferred to the items standing on the container floor. Then, the weight of the next item in J_i^{LBS} is transferred downwards. This procedure is repeated for all items in J_i^{LBS}.

The framework for checking compliance with the load bearing strength constraint is outlined in Algorithm 4.13. First, the procedure is initialized by determining J_i^{LBS} and J_j^{dsupp} for the all relevant items, and by setting the loads carried by each unit

area on the top faces of each item to 0. The sets J_j^{dsupp} of all relevant items are stored in set J^{dsupp}.

Algorithm 4.13 Testing for compliance with the load bearing strength constraint

Input: current item I_i with placement Π_i', set of already packed items and their placements K
Output: result of the test $feas$

```
1: procedure CheckLoadBearing
2:    determine J_i^LBS                          ▷ item set to be considered for this test
3:    J^dsupp := ∅                                                          ▷ initialization
4:    for each I_j ∈ J_i^LBS do
5:       determine J_j^dsupp                          ▷ items directly supporting I_j
6:       J^dsupp := J^dsupp ∪ {J_j^dsupp}
7:       load_j^unit(ua) := 0, ∀ unit areas ua on the top face of I_j     ▷ global variables
8:    end for
9:    for each item I_j ∈ J_i^LBS do
10:      feas := CheckLB_Item(I_j, J^dsupp, d_j, K)                    ▷ Algorithm 4.14
11:      if feas := false then
12:         break          ▷ violation of the load bearing strength constraint
13:      end if
14:   end for
15: end procedure
```

Then, the procedure CheckLB_Item, which is outlined in Algorithm 4.14, is called on each item $I_j \in J_i^{LBS}$. It serves to transmit the load, i.e. the weight of I_j, towards all items *directly* below I_j. The variable *load*, which is passed to the procedure, refers to the weight of I_j if the procedure is called by CheckLoadBearing. Later, recursive calls of the procedure are conducted. Then, *load* refers to the (partial) weight that is transmitted from an item placed above.

Algorithm 4.14 Transmitting a load to lower items

Input: current item I_j, J^{dsupp}, *load*, K
Output: result of the test $feas$

```
1: procedure CheckLB_Item
2:    feas := true
3:    for each I_k ∈ J_j^dsupp do                    ▷ J_j^dsupp is stored in J^dsupp
4:       feas := CalcLoad(I_j, I_k, load, J^dsupp, K)                  ▷ Algorithm 4.15
5:       if feas := false then
6:          break          ▷ violation of the load bearing strength constraint
7:       end if
8:    end for
9: end procedure
```

For each item I_k placed directly below the current item I_j, the part of *load* is determined that is transferred from I_j to I_k. This is done by calling the procedure

CalcLoad (Algorithm 4.15).

Algorithm 4.15 Calculation of the transmitted load

Input: upper item I_j, lower item I_k, *load* ((additional) bearing load transmitted from I_j to I_k), J^{dsupp}, K

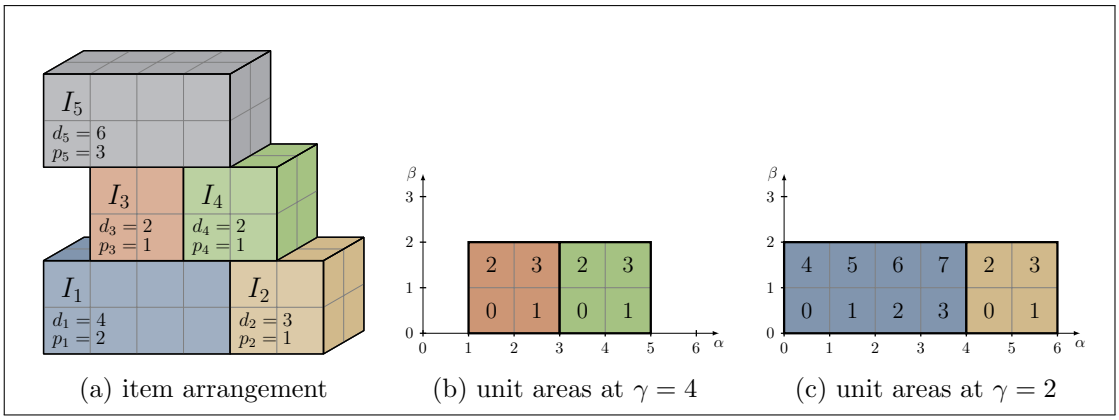
Output: result of the test $feas$, $load_k^{unit}(ua)$ $\forall$ unit areas ua on the top face of I_k

```
1: procedure CalcLoad
2:    feas := TRUE
3:    determine a_j                                  ▷ supported base area of I_j
4:    determine supp_jk                  ▷ base area of I_j that is supported by I_k
5:    ps := supp_jk / a_j                                  ▷ percentage support
6:    load_jk := load · ps          ▷ proportional load transmitted from I_j to I_k
7:    lpu := load_jk / supp_jk                              ▷ load per unit area
8:    if lpu > p_k then
9:       feas := FALSE
10:      goto End
11:   end if
12:   for each unit area ua of I_k where I_j is supported do
13:      load_k^unit(ua) := load_k^unit(ua) + lpu
14:      if load_k^unit(ua) > p_k then
15:         feas := FALSE
16:         goto End
17:      end if
18:   end for
19:   feas :=CheckLB_Item(I_k,J^dsupp, load_jk, K)
20:   End:
21: end procedure
```

First, $supp_{jk}$ is determined as the base area of I_j that is supported by I_k (cf. (4.15)). ps refers to the percentage of the supported bottom area of I_j that is supported by I_k $\left(ps = \frac{supp_{jk}}{a_j}\right)$ and $load_{jk}$ is the part of *load* that is transmitted from I_j to I_k ($load_{jk} = load \cdot ps$). Based on this, the load lpu that is transferred on each unit area of the touching area can be determined $\left(lpu = \frac{load_{jk}}{supp_{jk}}\right)$. If lpu itself exceeds the load bearing strength of I_k, the constraint is violated and the procedure is aborted. Otherwise, lpu is added to the current load on each unit area of I_k where I_j and I_k touch, and it is checked whether the load bearing strength of I_k is exceeded at any point. Subsequently, the recursive procedure call of CheckLB_Item is conducted. That is, the load that was transferred from I_j to I_k is further transferred to the items below I_k.

The whole procedure is illustrated by means of an example (Figure 4.12). An item stack is depicted in Figure 4.12a where the weights and load bearing strengths of all items are given, and I_5 is the newly placed item. Moreover, the numbering of the unit areas on the top faces of the items below is illustrated in Figures 4.12b

and 4.12c. Let the sequence in I_5^{LBS} be: $I_5^{LBS} = \{I_5, I_3, I_4, I_1, I_2\}$.[20] That is, the load transmission from I_5 towards the items directly below it, is considered first.



(a) item arrangement (b) unit areas at $\gamma = 4$ (c) unit areas at $\gamma = 2$

Figure 4.12: Example arrangement of items and numbering of unit areas for determining the bearing load

The call sequence of the procedures is illustrated in Figure 4.13 and continued in Figure 4.14.

First (Figure 4.13), the overall procedure is initialized (cf. Algorithm 4.13, lines 2 to 8) and the weight of item I_5 is distributed towards the items directly supporting it (I_3 (arc 2), I_4 (arc 8)) and towards the items directly supporting those (I_1 (arcs 3ff.), I_2 (arcs 9ff.)). The calculated changes in the loads carried by the individual unit areas of the items are also illustrated (bold, red). Followingly (Figure 4.14), the weights of I_3 (arcs 18ff.) and I_4 (arcs 23ff.) are transmitted downwards and added to the bearing loads determined before. In the end, the procedure CheckLB_Item is also called with $I_i = I_1$ and $I_i = I_2$, respectively, because they are members of set I_5^{LBS}. However, the procedure would immediately return to CheckLoadBearing as there are no further items below I_1 and I_2 (arcs 31ff.).

For example, item I_1 bears the (partial) weights of items I_3 and I_5 on four of its unit areas ua ($ua \in \{1, 2, 5, 6\}$, cf. Figure 4.12c). I_5 is supported in six out of eight unit areas. That is, $\frac{1}{6}$ of the weight of I_5 (d_5) is transmitted per supported unit base area. $\frac{1}{8}$ of d_5 would be transmitted per unit area if the item was fully supported. Four unit areas are supported by I_3. Thus, I_3 carries two thirds of the weight of I_5, which equals 4 weight units (WU). As I_3 is completely supported by I_1, those 4 WU are equally spread over its bottom face and transmitted to I_1. Thus, the unit areas $ua \in \{1, 2, 5, 6\}$ each carry 1 WU of the weight of I_5. In addition, the weight of I_3

[20] The order of items in this set is not important for the procedure itself. It is just stated as a basis for the following remarks.

is also transferred to those unit areas, which equals 0.5 WU per unit area. Hence, the transferred weights from I_3 and I_5 add up to 1.5 WU per unit area on the top face of I_1. On two of the remaining unit areas $ua \in \{3, 7\}$ of I_1, the partial weights of items I_4 and I_5 are carried. I_4 carries one third of the weight of I_5. Those 2 WU are equally distributed over the bottom face of I_4 since it is completely supported, and the unit areas $ua \in \{3, 7\}$ of I_1 each bear 0.5 WU of this load. Finally, I_4 also transmits its own weight towards I_1, which leads to a total bearing load on those unit areas of 1 WU.

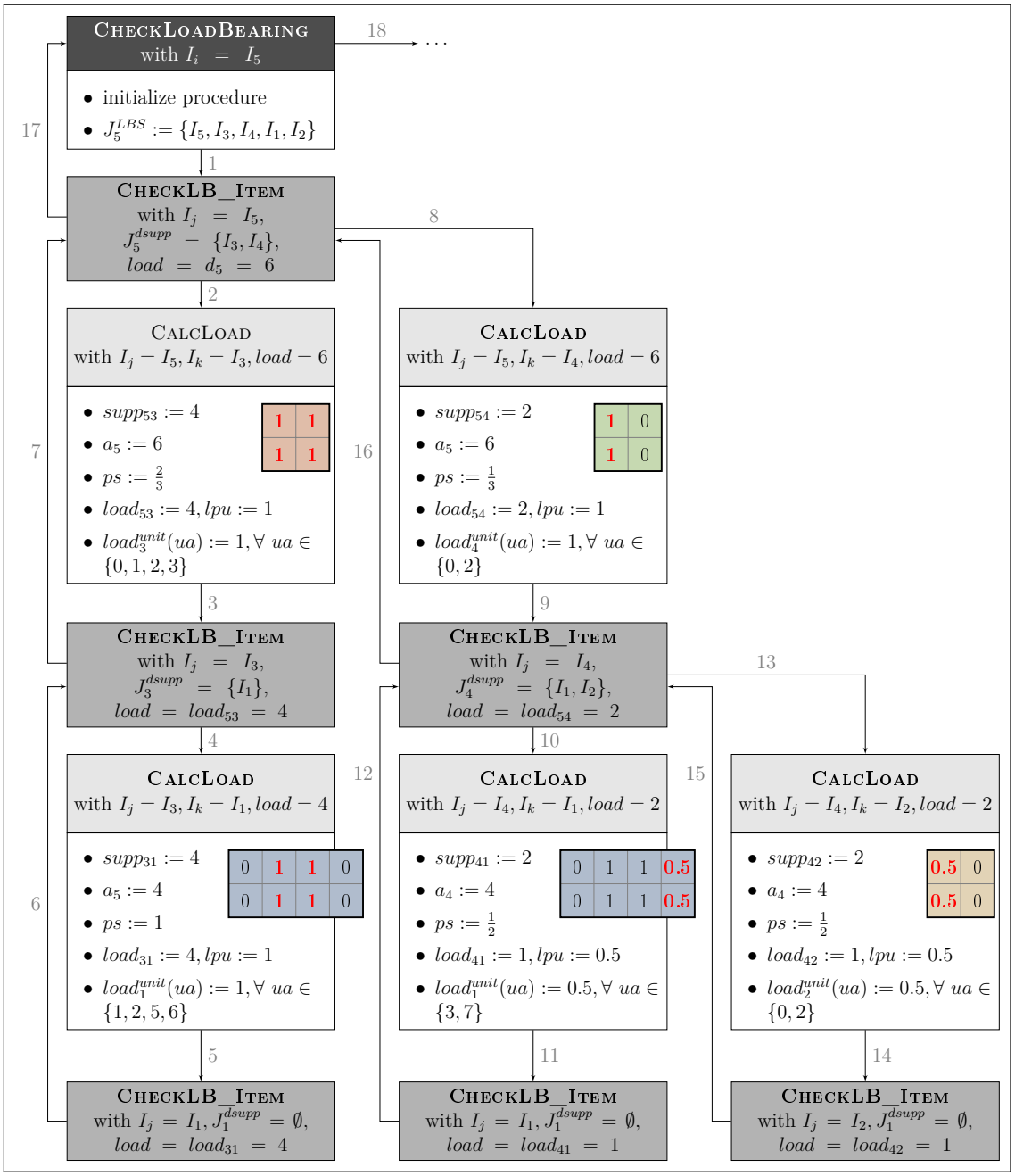


Figure 4.13: Exemplary call sequence for determining the bearing loads

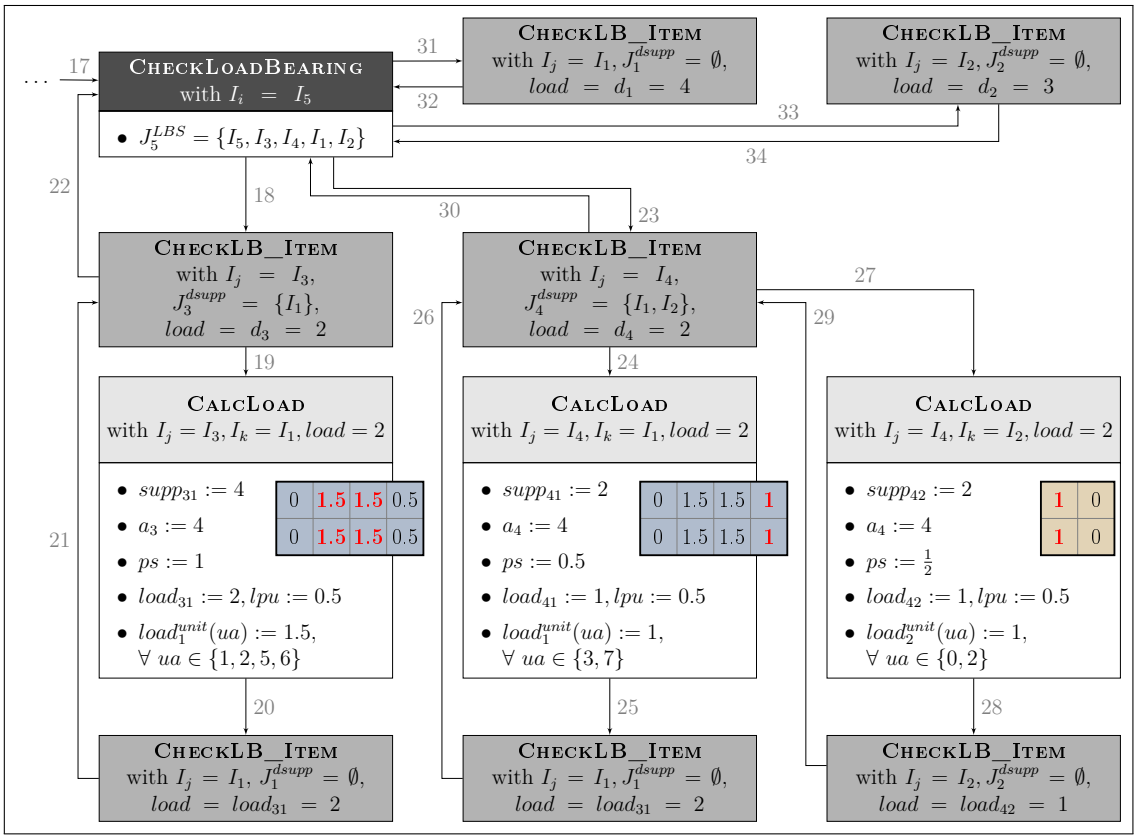


Figure 4.14: Exemplary call sequence for determining the bearing loads (*continued*)

4.2 Routing heuristics

Simple routing construction heuristics can be applied for solving a VRP very quickly and find reasonable solutions. Alternatively, more sophisticated procedures can be used in order to obtain routing solutions of higher quality. In this chapter, different routing heuristics are presented in order to solve a 3L-VRPBTW.

4.2.1 Savings heuristic

The well-known savings heuristic (Clarke and Wright, 1964) is a fast heuristic, is easy to implement and delivers good results. The basic idea is to realize connections of two customers that lead to high cost savings. The procedure is outlined in Algorithm 4.16.

Initially, each customer forms a direct trip, a so-called out-and-back tour.[21] For each arc connecting two customers $(i, j) \in E$ $(i, j > 0)$ the respective saving sav_{ij} is computed as follows:

$$sav_{ij} = c_{i0} + c_{0j} - c_{ij}. \tag{4.19}$$

[21] The terms *route* and *tour* can mostly be used interchangeably if the sequence of customers is unimportant to the remarks given. A route refers to a sorted sequence of customers, whereas a tour is an unsorted set of all customers visited in a route.

As the costs between two customers are assumed to be symmetric (see Chapter 2.1), only one saving per customer pair is required. The savings are sorted in a non-increasing order and, beginning with the highest saving, the algorithm tries to connect the respective two customers until no further saving can be realized. Two customers can be joined if:

(a) they are not already in the same tour,

(b) both customers still have at least one connection to the depot,

(c) capacity, time window and packing constraints (and sequencing constraints in the case of the 3L-VRPCB(TW)) are not violated in the newly formed route.

Let DC_i be a counter of the connections of customer i to the depot, vno_i be the vehicle (or tour) number, respectively, to which customer i is assigned and R'_v be an ordered set of customers visited by vehicle v.[22]

The adherence to the capacity and packing constraints needs to be checked according to the respective problem variant. For the 3L-VRPTW and 3L-VRPCB(TW) it is sufficient to check them for the beginning of the route for the linehaul customers and the end of the route for the backhaul customers. For the variants 3L-VRPMB(TW) and 3L-VRPSDP(TW), however, some constraints need to be checked for every station of a route (cf. Chapter 2). Due to the capacity, packing and time window constraints the direction of a route is important and should be considered in the savings heuristic. That is, if two customers i and j are to be joined and the new route in which i precedes j violates one of the mentioned constraints, the other direction should be tested as well (cf. line 22) as it could be feasible. The procedure of connecting two routes is outlined in Algorithm 4.17.

In order to ensure the sequence constraint (2.95) of the 3L-VRPCB(TW), the approach of Deif and Bodin (1984) is applied. A counter is assigned to every route counting the connections of a linehaul to a backhaul customer. This counter must not exceed a value of 1 for any route and no backhaul customer must precede a linehaul customer. Apart from that, the procedure as outlined above can be applied.

[22] Note that, though every route starts and ends at the depot, it is not included in the sequences R'_v here (as opposed to R_v introduced in Chapter 2).

Algorithm 4.16 Savings heuristic for a (3L-)VRPBTW

Input: instance data
Output: solution for a (3L-)VRPBTW with objective function value z

procedure SAVINGS
  for each customer $i \in N \setminus \{0\}$ **do** ▷ initialization
    $DC_i := 2$, $vno_i := i$ ▷ create a separate route for each customer
    $R'_{vno_i} := \{i\}$
  end for
  $z := 2 \cdot \sum_{i \in N_c} c_{i0}$ ▷ objective function value: total cost of all routes
  $savList := \emptyset$
  for each arc $(i, j) \in E$ $(i, j > 0)$ **do**
    $sav_{ij} = c_{i0} + c_{0j} - c_{ij}$ ▷ calculate savings
    $savList := savList \cup \{(sav_{ij}, i, j)\}$
  end for
  sort all savings in list $savList$ in a non-increasing order
  while $savList \neq \emptyset$ **do**
    $sav := savList(1)$ ▷ highest available saving in $savList$
    $i, j :=$ customers belonging to sav
    if $DC_i > 0$ **and** $DC_j > 0$ **and** $vno_i \neq vno_j$ **then** ▷ restrictions (a) and (b)
      $R''_i := R'_{vno_i}$ with customer i at the end ▷ temporary routes
      $R''_j := R'_{vno_j}$ with customer j at the beginning
      $R'' := R''_i \cup R''_j$
      if R'' satisfies (c) **then** ▷ see restrictions listed above
        CONNECT$(R'', vno_k\ \forall k \in R'', i, j, DC_i, DC_j, R'_{vno_i}, R'_{vno_j}, sav_{ij}, z)$
        ▷ Algorithm 4.17
      else ▷ try the other direction
        reverse R''
        if R'' satisfies (c) **then**
          CONNECT$(R'', vno_k\ \forall k \in R'', i, j, DC_i, DC_j, R'_{vno_i}, R'_{vno_j}, sav_{ij}, z)$
        end if
      end if
    end if
    $savList := savList \setminus \{sav\}$
  end while
end procedure

Algorithm 4.17 Connecting two customers in the course of the savings heuristic

Input: new route R'', vehicle numbers $vno_k \forall k \in R''$, connected customers i and j, $DC_i, DC_j, R'_{vno_i}, R'_{vno_j}, sav_{ij}$, current objective function value z
Output: updated solution data and objective function value z

procedure CONNECT
  $R'_{vno_i} := R'', R'_{vno_j} := \emptyset$ ▷ connect routes vno_i and vno_j
  for each $k \in R'_{vno_i}$ **do**
    $vno_k := vno_i$ ▷ update tour numbers
  end for
  $DC_i := DC_i - 1$ ▷ update depot connections
  $DC_j := DC_j - 1$
  $z := z - sav_{ij}$ ▷ update objective function value
end procedure

4.2.2 Adaptive large neighbourhood search

In order to obtain solutions of high quality, a metaheuristic is used for solving the 3L-VRPBTW. Namely, an adaptive large neighbourhood search (ALNS) algorithm is applied that is based on the ALNS presented by Ropke and Pisinger (2006a,b).
The large neighbourhood search (LNS) was first proposed by Shaw (1997, 1998) and applied to the CVRP and the VRPTW. In every iteration of the heuristic, customers are removed from the current solution. For that purpose, a heuristic is employed determining the customers to be removed. Subsequently, they are reinserted into the solution. For the reinsertion procedure, a branch-and-bound (B&B) approach is applied by Shaw (1997, 1998). If the newly generated solution is better than the starting solution (of the previous iteration) from which it resulted, it is used as the starting solution for the next iteration. Otherwise, the previous starting solution is used again.
The ALNS is a further development of the LNS. One main difference is that more than one heuristic is available for both the removal and the insertion procedures, while an "expensive" B&B approach is not employed for insertions. Three different removal heuristics are used by Ropke and Pisinger (2006b) and six by Ropke and Pisinger (2006a). Further removal heuristic are, e.g., presented by Demir et al. (2012). Moreover, Ropke and Pisinger (2006a,b) use various rather simple, greedy insertion heuristics. In the course of this thesis, it is attempted to equip the heuristic with an even larger number of heuristics. The ones used here are described in detail in Chapters 4.2.2.1 and 4.2.2.2. Furthermore, the integration of solving a packing problem within the ALNS is demonstrated in Chapter 4.2.2.3.
Another difference of the approach of Ropke and Pisinger (2006a,b) compared to the original LNS, is to embed the ALNS into a simulated annealing framework that controls the acceptance of new solutions. That is, a solution that is worse than the previous solution can be accepted and be the new starting solution with a certain probability. The acceptance procedure is explained in Chapter 4.2.2.4.
Furthermore, in every iteration the number of customers to be removed (n_{rem}), the removal and the insertion heuristics are chosen randomly. n_{rem} is selected from a given interval. A specific feature of the ALNS is that the probability of choosing a certain heuristic changes throughout the search and depends on its performance in previous iterations. (Therefore, the additional term "adaptive" is used.) This aspect

is further described in Chapter 4.2.2.5. Finally, the applied termination criteria are listed in Chapter 4.2.2.6.

The overall ALNS procedure for a (3L-)VRPBTW is outlined in Algorithm 4.18.[23]

Algorithm 4.18 Adaptive large neighbourhood search for a (3L-)VRPBTW

Input: instance data, parameters
Output: solution for a (3L-)VRPBTW s_{best} with objective function value $z(s_{best})$

procedure ALNS
  construct initial solution s_{init}
  $s_{curr} := s_{best} := s_{init}$
  while stopping criterion is not met **do**
    select number of customers to be removed n_{rem}
    select removal heuristic REM and insertion heuristic INS
    determine next solution $s_{next} :=$INS(REM(s_{curr}, n_{rem}))
    check acceptance of s_{next}
    if s_{next} is accepted **then**
      $s_{curr} := s_{next}$
      if $z'(s_{curr}) < z'(s_{best})$ **then** ▷ $z'(s)$: total cost of solution s (see Chapter 4.2.2.4)
        $s_{best} := s_{curr}$
      end if
    end if
    if segment end is reached **then** ▷ i.e. after a certain number of iterations
      update weights of insertion and removal heuristics
    end if
  end while
end procedure

4.2.2.1 Removal heuristics

In total, 21 different removal heuristics are implemented and tested, some of which are based on heuristics proposed in the literature and some have been newly developed. The heuristics with their corresponding sources are listed in Table 4.2 (*new* refers to a newly developed heuristic). In the final hybrid algorithm, nine of the heuristics are used (cf. Chapter 5.3), which is also indicated in Table 4.2. In the following, the used heuristics are explained in greater detail. Subsequently, short descriptions of the omitted heuristics are provided.[24]

Some heuristics require quite specific solution structures. It may happen that a solution does not contain those structure (overlapping routes or intersections within routes). In this case, the respective removal heuristic cannot be applied and another

[23] The instance data and parameters are assumed to be provided in the following pseudocodes.
[24] For further details, please refer to the respective sources.

heuristic is chosen instead.

Table 4.2: Overview of removal heuristics

name	**source**	**in final ALNS**
Shaw removal	Shaw (1997)*	✓
random removal	Ropke and Pisinger (2006b)	✓
worst removal	Ropke and Pisinger (2006b)	✓
cluster removal	Ropke and Pisinger (2006a)	✓
neighbour graph removal	Ropke and Pisinger (2006a)	✓
overlap removal	*new*	✓
inner route removal	*new*	✓
intersection removal	*new*	✓
route pair removal	*new*	✓
random-route removal	Bortfeldt et al. (2015)	
least customer-route removal	*new*	
average distance-route removal	*new*	
largest distance-route removal	*new*	
proximity-based Shaw removal	Demir et al. (2012)	
time-based Shaw removal	Demir et al. (2012)	
demand-based Shaw removal	Demir et al. (2012)	
worst distance removal	Demir et al. (2012)	
worst time removal	Demir et al. (2012)	
historical knowledge removal	Demir et al. (2012)	
average distance removal**	Demir et al. (2012)	
node neighbourhood removal	Demir et al. (2012)	

*: The heuristic is not adopted unchanged, but is adapted to the (3L-)VRPBTW.
**: In Demir et al. (2012), it is called *neighbourhood removal*. The operator is renamed here in order to avoid confusion with other, similarly named heuristics.

Shaw removal

The Shaw removal heuristic is based on the original removal heuristic introduced by Shaw (1997, 1998). Some components are adopted from Ropke and Pisinger (2006b), Demir et al. (2012) and Bortfeldt et al. (2015), and the heuristic is adjusted to the (3L-)VRPBTW. The idea is to remove similar customers from a solution because it appears to be easier to exchange similar customers. In contrast, removing very different customers might lead to a next solution that is almost identical to the current one as the removed customers cannot take new positions (Ropke and Pisinger, 2006b). The degree of similarity is defined by a relatedness value that is determined in the following manner: Let $netvol_i$ be the net demand volume of customer i. In the case of the 3L-VRPTW, the net volume of a customer equals the total demanded volume:

$$netvol_i = \sum_{k=1}^{m_i} (l_{ik} \cdot w_{ik} \cdot h_{ik}), \forall\ i \in N_c. \tag{4.20}$$

The same applies to the 3L-VRPCB(TW) and 3L-VRPMB(TW). However, backhaul customers are assumed to have a negative net volume:

$$netvol_i = \begin{cases} \sum_{k=1}^{m_i}(l_{ik} \cdot w_{ik} \cdot h_{ik}), & \forall\ i \in N^L \\ -\sum_{k=1}^{m_i}(l_{ik} \cdot w_{ik} \cdot h_{ik}), & \forall\ i \in N^B. \end{cases} \tag{4.21}$$

In the case of the 3L-VRPSDP(TW), the net volume represents the difference between the delivery and pickup demand volume:

$$netvol_i = \sum_{k=1}^{m_i^L}(l_{ik} \cdot w_{ik} \cdot h_{ik}) - \sum_{k=m_i^L+1}^{m_i}(l_{ik} \cdot w_{ik} \cdot h_{ik}), \forall\ i \in N_c. \tag{4.22}$$

The net volumes are normalized ($netvol_i^*$) in the range [0,1]. That is,

$$netvol_i^* = \frac{netvol_i - netvol_{min}}{netvol_{max} - netvol_{min}}, \forall\ i \in N_c, \tag{4.23}$$

where $netvol_{min}$ and $netvol_{max}$ refer to the minimum and maximum net demand volumes among all customers:

$$netvol_{min} = \min_{i \in N_c} netvol_i, \tag{4.24}$$

$$netvol_{max} = \max_{i \in N_c} netvol_i. \tag{4.25}$$

Furthermore, let c_{ij}^* be the normalized cost (in the range [0,1]) of the directed edge $(i,j) \in E$ and RT_i^* be the normalized ready-time (in the range [0,1]) of customer i ($i \in N_c$). The normalized costs and ready times are determined analogously as the normalized demand volume (cf. (4.23)).

In addition, a binary variable $\varepsilon_{\mathrm{ij}}$ describes whether two customers i and j ($i, j \in N_c$) are in the same tour ($\varepsilon_{ij} = 1$) or not ($\varepsilon_{ij} = 0$). Let $\omega_1, \omega_2, \omega_3$ and ω_4 be predefined weights for the calculation of the relatedness value.[25] The relatedness $relate(i,j)$ of two customers i and j ($i, j \in N_c$) can then be calculated as:

$$relate(i,j) = \omega_1 \cdot c_{ij}^* + \omega_2 \cdot \left|RT_i^* - RT_j^*\right| + \omega_3 \cdot \left|netvol_i^* - netvol_j^*\right| + \omega_4 \cdot \varepsilon_{ij}. \tag{4.26}$$

The procedure of the Shaw removal operator is shown in Algorithm 4.19. Initially, one customer is randomly selected and added to the set of customers that are re-

[25] If no time windows are considered, ω_2 equals 0.

moved in the end (Rem). In every iteration of the operator, i.e. until n_{rem} customers are added to Rem, one customer from Rem is randomly chosen and another customer that is closely related to it is added to the set. In the course of this, not necessarily the customer with the highest relatedness measure is chosen (lines 13 and 14). Some randomness is introduced to the selection by means of the determinism parameter ρ ($\rho \geq 1$) (cf. Ropke and Pisinger, 2006b). A low value of ρ refers to much randomness, and vice versa.

Algorithm 4.19 ALNS: Shaw removal heuristic (Adapted from Ropke and Pisinger, 2006b)

Input: s_{curr}, n_{rem}
Output: s_{next}

```
 1: procedure ShawRemoval
 2:     s_next := s_curr
 3:     randomly select customer i
 4:     Rem := {i}                          ▷ set of customers to be removed
 5:     while |Rem| < n_rem do
 6:         randomly select a customer i ∈ Rem
 7:         S := ∅
 8:         for each customer j ∈ N_c \ Rem do
 9:             calculate relatedness measure relate(i, j)
10:             S := S ∪ {j}
11:         end for
12:         sort S by ascending relate(i, j)
13:         choose a random number y ∈ [0, 1)
14:         Rem := Rem ∪ {k} with k := S[y^ρ · |S|]
15:     end while
16:     remove the customers in Rem from s_next
17: end procedure
```

Random removal

Probably the simplest removal heuristic is the random removal heuristic. As the name suggests, all removed customers are determined completely randomly. The procedure is depicted in Algorithm 4.20. The set of planned customers S refers to the customers that are assigned to a route in the current solution s_{curr}. As some customers can be unassigned (they are contained in the set of missing customers; see below) not necessarily all n customers are included in S.

As Ropke and Pisinger (2006b) point out, this operator would be a special case of the above presented Shaw heuristic with $\rho = 1$, but it is of course more efficient to implement the random removal heuristic separately.

Algorithm 4.20 ALNS: random removal heuristic

Input: s_{curr}, n_{rem}
Output: s_{next}
1: **procedure** RANDOMREMOVAL
2: $s_{next} := s_{curr}$
3: $S :=$ set of planned customers in s_{curr}
4: $Rem := \emptyset$ ▷ set of customers to be removed
5: **while** $|Rem| < n_{rem}$ **do**
6: randomly choose a customer $i \in S$
7: $Rem := Rem \cup \{i\},\ S := S \setminus \{i\}$
8: **end while**
9: remove the customers in Rem from s_{next}
10: **end procedure**

Worst removal

Customers are removed that supposedly deteriorate the solution the most. In this context, the *cost* of a customer is defined as the difference between the total cost of the current solution and the total cost of the solution if the customer was completely removed. In the course of this, only the differences in the resulting routing costs, i.e. the total travel distances or times (depending on whether time windows are considered or not; see Chapter 2.1), are regarded. Penalty costs for missing customers are not considered (see Chapter 4.2.2.4). The whole procedure is depicted in Algorithm 4.21. Randomness is introduced into the selection of a customer as before (lines 6 and 7). Furthermore, unlike in the operators presented above, the temporary solution s_{next} is updated after every removal, i.e. the costs also need to be recalculated after every removal.

Algorithm 4.21 ALNS: worst removal heuristic (Adapted from Ropke and Pisinger, 2006b)

Input: s_{curr}, n_{rem}
Output: s_{next}
1: **procedure** WORSTREMOVAL
2: $s_{next} := s_{curr}$
3: **while** $n_{rem} > 0$ **do**
4: determine set of planned customers S (in s_{next})
5: sort the customers in S by descending $cost(i, s_{next})$
6: choose a random number $y \in [0, 1)$
7: $i := S\left[y^{\rho} \cdot |S|\right]$
8: remove i from s_{next} ▷ s_{next} is updated
9: $n_{rem} := n_{rem} - 1$
10: **end while**
11: **end procedure**

Cluster removal

In the cluster removal heuristic, one route is initially selected randomly. This route is then divided into two clusters by (partly) solving a minimum spanning tree problem with a modified variant of the algorithm of Kruskal (1956). Originally, in every iteration of Kruskal's algorithm the arc that is associated with the lowest cost is selected. This arc must not have been selected before and must not lead to a circle with previously selected arcs. The procedure is repeated until the selected arcs form a spanning tree. In the modified version considered here, the algorithm is run until two connected clusters (trees) are generated.

Subsequently, one of the generated clusters is chosen randomly and the included customers are added to Rem, the set of customers to be removed. If more customers are required for removal, a customer i is picked from Rem and another customer is determined that is closest to i and is not included in a route that was affected by the clustering before. The route of this customer is then clustered and the process described above is continued until (at least) n_{rem} customers have been removed. The pseudocode for the cluster removal operator is provided in Algorithm 4.22.

Algorithm 4.22 ALNS: cluster removal heuristic

Input: s_{curr}, n_{rem}
Output: s_{next}

1: $s_{next} := s_{curr}$
2: partition a randomly selected route v into two clusters Cl_1 and Cl_2 using a modified KRUSKAL's algorithm
3: randomly select one cluster $Cl \in \{Cl_1, Cl_2\}$
4: $Rem := Cl$ ▷ set of customers to be removed
5: $C := \{v\}$ ▷ set of routes affected by clustering
6: **while** $|Rem| < n_{rem}$ **do**
7: randomly select a customer i from Rem
8: find a customer that is closest to i from a route $v \notin C$
9: $C := C \cup \{v\}$
10: partition route v into two clusters Cl_1 and Cl_2 ▷ as above
11: randomly select one cluster $Cl \in \{Cl_1, Cl_2\}$
12: $Rem := Rem \cup Cl$
13: **end while**
14: remove the customers in Rem from s_{next}

An example where the cluster removal could be useful is depicted in Figure 4.15. Route 2 is divided into two clusters, which are highlighted in Figure 4.15a. One cluster (customers 8-11) is removed from route 2. Possible insertions could then be made into route 1 (Figure 4.15b).

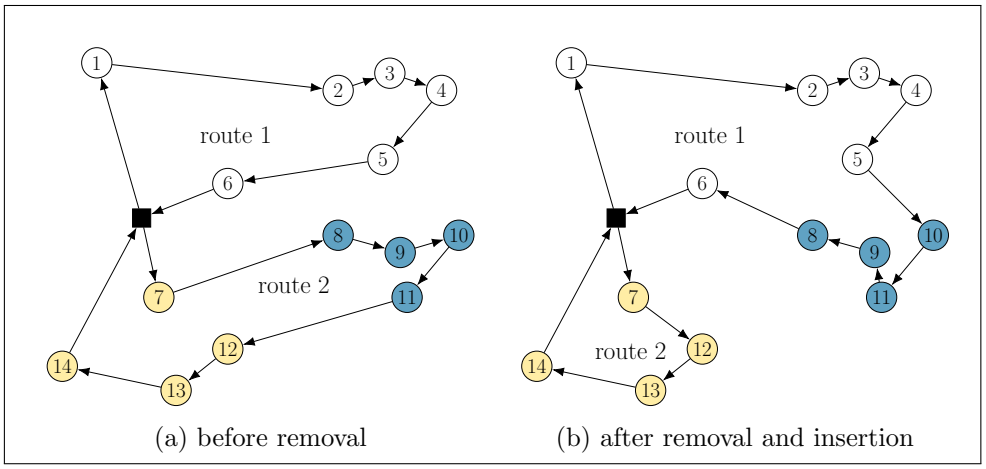


Figure 4.15: Example for a cluster removal (Adapted from Ropke and Pisinger, 2006a)

Neighbour graph removal

The neighbour graph removal operator makes use of historical information (Ropke and Pisinger, 2006a). For this purpose, a neighbour graph $NG = (N_{NG}, E_{NG})$ with the node set N_{NG} and the (directed) edge set E_{NG} is employed. The graph is complete, directed and weighted. Each node in N_{NG} represents one customer and the cost c_{ij}^{NG} of a direct edge $(i,j) \in E_{NG}$ represents the cost of the best solution found so far in which customer i is visited directly before customer j in the same route. At the beginning of the ALNS, the c_{ij}^{NG} of all arcs $(i,j) \in E_{NG}$ are initialized with sufficiently large values.

The detailed procedure is outlined in Algorithm 4.23.

Algorithm 4.23 ALNS: neighbour graph removal heuristic

Input: s_{curr}, n_{rem}, NG
Output: s_{next}

procedure NeighbourGraphRemoval
  $s_{next} := s_{curr}$
  $S :=$ set of planned customers in s_{curr}
  $Rem := \emptyset$ ▷ set of customers to be removed
  for each customer $j \in S$ **do** ▷ calculate score
    $sc_j = c_{ij}^{NG} + c_{jk}^{NG}$ ▷ i, k: j's predecessor and successor
  end for
  sort S by non-increasing scores
  while $|Rem| < n_{rem}$ **do**
    choose a random number $y \in [0,1)$
    $i := S\,[y^{\rho} \cdot |S|]$
    $S := S \setminus \{i\},\ Rem := Rem \cup \{i\}$
  end while
  remove the customers in Rem from s_{next}
end procedure

In order to determine the customers to be removed, a score is calculated for each

customer j by summing up the costs of the two directed edges in NG that describe his current position. Let i be the predecessor of j in the current solution and k be the successor of j. The score assigned to j equals the sum of the cost of the directed edges (i, j) and (j, k) in NG. A high score indicates an unsuitable placement, i.e. comparatively bad previous solutions with the constellation $(\dots, i, j, k, \dots)$ in a route. Thus, customers with high scores are removed. The neighbour graph is updated every time a new solution is found.

Overlap removal

This operator aims at eliminating intersections between two routes. Two routes intersect or overlap if at least one travelled arc of one route is crossing a travelled arc of the other. The heuristic removes customers in the overlapping area. The procedure is depicted in Algorithm 4.24.

Algorithm 4.24 ALNS: overlap removal heuristic

Input: s_{curr}, n_{rem}
Output: s_{next}

procedure OverlapRemoval
  $s_{next} := s_{curr}$
  $S :=$ all pairs of overlapping routes
  if $S \neq \emptyset$ **then** ▷ the heuristic can only be applied if there is at least one pair of overlapping routes
    $Rem := \emptyset$ ▷ set of customers to be removed
    while $|Rem| < n_{rem}$ **and** $|S| > 0$ **do**
      randomly select a route pair $rp \in S$
      $S := S \setminus \{rp\}$
      $O :=$ set of customers in overlapping area of rp
      while $|Rem| < n_{rem}$ **and** $|O| > 0$ **do**
        randomly select a customer $i \in O$
        $O := O \setminus \{i\}$
        if $i \notin Rem$ **then**
          $Rem := Rem \cup \{i\}$
        end if
      end while
    end while
    remove the customers in Rem from s_{next}
    if $|Rem| < n_{rem}$ **then** ▷ no more overlapping routes available
      RandomRemoval$(s_{next}, n_{rem} - |Rem|)$
    end if
  end if
end procedure

Let (cx_i, cy_i) be the coordinate pair representing the position of customer i in a two-dimensional Cartesian coordinate system. It begins with the determination of

all overlapping route pairs (line 3). This is done in the following way: Each route is represented by a rectangle formed by the minimum and maximum *cx*- and *cy*-coordinates of the locations visited in the route (including the depot). If those rectangles of two routes overlap, it is checked whether there is any pair of travelled arcs of the respective routes (one arc per route) that is intersecting.

Examples are depicted in Figure 4.16. The grey-shaded areas represent the overlap of the rectangular areas formed by the routes (dashed lines). In Figure 4.16a, the arcs (3,4), (4,5), (5,6), (6,0), (0,7), (7,8), (8,9), (9,10) and (11,0) are checked for intersections because they are (at least partly) within the overlapping area. As the arcs (3,4) and (8,9) intersect, the two routes overlap. In Figure 4.16b, the routes also form overlapping rectangles. However, their are no intersecting arcs of the routes in this area.

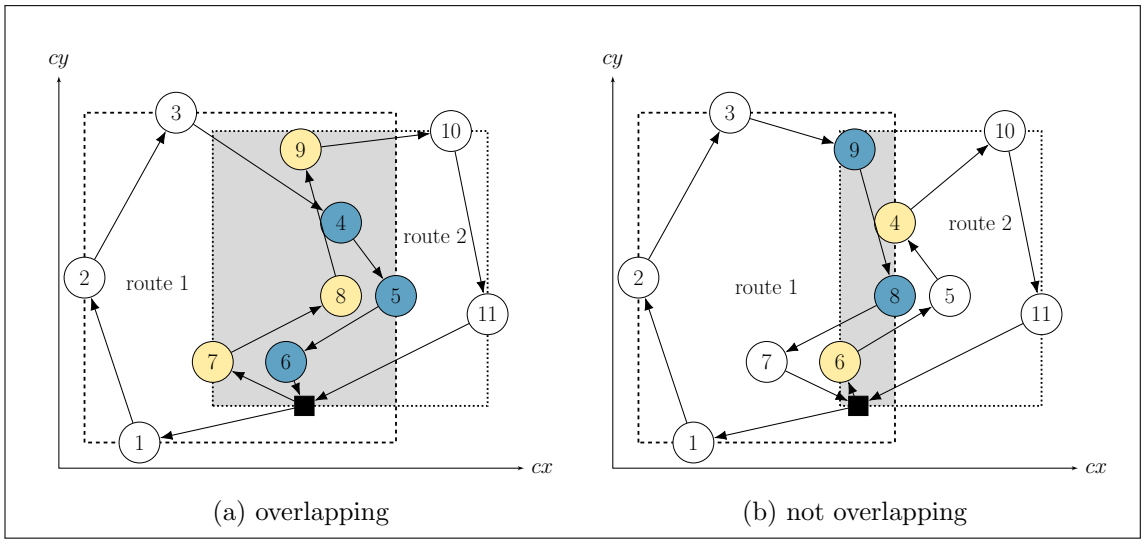


Figure 4.16: Examples for overlapping and non-overlapping routes

After this procedure, overlapping route pairs are selected randomly and customers from the overlapping area are removed from the solution (Algorithm 4.24, line 6ff.). If there are not enough intersecting route pairs, the missing removal customers are chosen randomly out of all customers that remain in the solution (line 19).

If there are no intersecting route pairs at all in the current solution, another removal heuristic is chosen. For this purpose, the higher-level function checks whether the current solution was changed by the removal heuristic.

Inner route removal

The inner route removal heuristic does not only remove customers from a solution, but it also changes the given structure of routes. It aims at removing short routes

that are enclosed in larger ones, and splits up the large routes. The whole procedure is outlined in Algorithm 4.25.

Algorithm 4.25 ALNS: inner route removal heuristic

Input: s_{curr}, n_{rem}
Output: s_{next}

procedure InnerRouteRemoval
  $s_{next} := s_{curr}$
  $count_{rem} := 0$ ▷ number of removed customers
  $S :=$ all *inner route-outer route* pairs
  if $S \neq \emptyset$ **then**
    set of affected routes $AR := \emptyset$
    while $count_{rem} < n_{rem}$ **do**
      randomly select a route pair $rp \in S$
      $v_{in} :=$ index of the inner route of rp
      $n_{v_{in}} :=$ number of customers in route $R_{v_{in}}$
      $v_{out} :=$ index of the outer route of rp
      if $v_{in} \notin AR$ **and** $v_{out} \notin AR$ **then**
        $AR := AR \cup \{v_{in}, v_{out}\}$
        $count_{rem} := count_{rem} + n_{v_{in}}$
        remove $R_{v_{in}}$ from the solution s_{next}
        split $R_{v_{out}}$, update s_{next}
      end if
      $S := S \setminus \{rp\}$
      if $count_{rem} < n_{rem}$ **and** $S = \emptyset$ **then** ▷ no more route pairs available
        RandomRemoval$(s_{next}, n_{rem} - count_{rem})$
        $count_{rem} := n_{rem}$
      end if
    end while
  end if
end procedure

An example is shown in Figure 4.17. As in the overlap removal heuristic, rectangles enclosing the routes are determined (Figure 4.17a). If such a rectangle is completely covered by another, the corresponding route is called an *inner route* (e.g. route $(0, 7, 8, 9, 10, 0)$ in Figure 4.17a), and the route related to the enclosing rectangle is called the *outer route*. A selected inner route is removed from the solution. Moreover, the related *outer route* is split into two routes (Figure 4.17b). The "cut" is made after the first $\lceil n_{v_{out}}/2 \rceil$ customers in the route, where $n_{v_{out}}$ is the number of customers in the outer route. Figure 4.17c shows how the solution could look like after the application of an insertion heuristic.

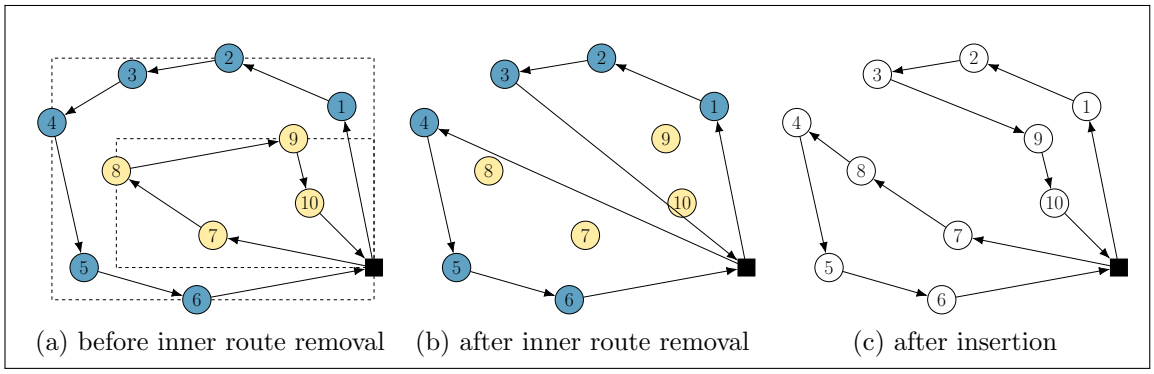


Figure 4.17: Example for an inner route removal

Intersection removal

This operator also aims at removing intersections. However, unlike the overlap removal heuristic, it focuses on intersections within individual routes. The heuristic procedure is depicted in Algorithm 4.26. At the beginning, all crossings within the routes are determined. Then, customers are removed that form the respective crossing arcs (excluding the depot).

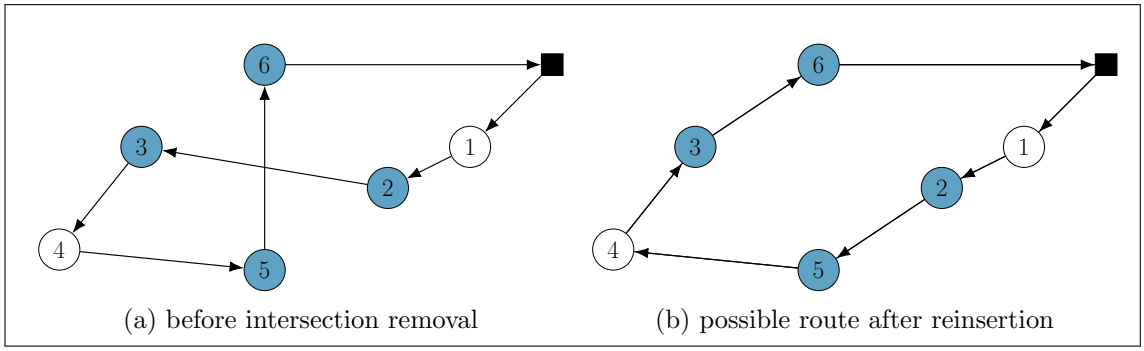
Algorithm 4.26 ALNS: intersection removal heuristic

Input: s_{curr}, n_{rem}
Output: s_{next}

procedure IntersectionRemoval
  $s_{next} := s_{curr}$
  $S :=$ all pairs of crossing arcs within each route
  if $S \neq \emptyset$ **then**
    $Rem := \emptyset$ ▷ set of customers to be removed
    while $|Rem| < n_{rem}$ **and** $|S| > 0$ **do**
      randomly select a crossing arc pair $cap := ((i_1, i_2), (i_3, i_4)), cap \in S$
      ▷ i_j: customers forming the arcs
      $S := S \setminus \{cap\}$
      for $j := 1$ **to** 4 **do**
        if $i_j \neq 0$ **and** $i_j \notin Rem$ **then**
          $Rem := Rem \cup \{i_j\}$
        end if
      end for
    end while
    remove the customers in Rem from s_{next}
    if $|Rem| < n_{rem}$ **then** ▷ no more arc pairs available
      RandomRemoval$(s_{next}, n_{rem} - |Rem|)$
    end if
  end if
end procedure

An example is illustrated in Figure 4.18a. Here, the crossing edges are (2,3) and (5,6). The respective customers would be removed from the solution. A possible

new route that could be built in the insertion process is depicted in Figure 4.18b.



(a) before intersection removal (b) possible route after reinsertion

Figure 4.18: Example for an intersection removal

Route pair removal

This operator removes two *complete* routes that are intersecting. The intersecting routes are determined as in the overlap removal operator. Thus, it could happen that more than n_{rem} customers are removed if the routes contain more. This case is accepted, though. On the other hand, if there are less than n_{rem} customers in the route, the remaining customers are removed randomly.

The following heuristics are also implemented and tested, but not applied in the final hybrid algorithm:

Shaw removal variants

These variations of the Shaw removal operator focus the calculation of the relatedness factor on only one aspect. That is, in each case only one weight is effective:

- proximity-based: removes customers that are related with respect to distance $(\omega_1 = 1, \omega_2 = \omega_3 = \omega_4 = 0)$,
- time-based: removes customers that are related with respect to time windows $(\omega_2 = 1, \omega_1 = \omega_3 = \omega_4 = 0)$,
- demand-based: removes customers that are related with respect to demand $(\omega_3 = 1, \omega_1 = \omega_2 = \omega_4 = 0)$.

Route removal variants

These operators remove entire routes from the solution. Similar to the route pair removal, at least n_{rem} customers are removed. Different variants of the operator are considered with different criteria for the routes to be removed:

- The removed route is chosen randomly out of all routes (*random-route removal*).
- The route with the least number of customers is removed (*least customer-route removal*). The chances are higher to assign all customers of the route to new routes if a very short route (with respect to the number of customers) is dissolved.
- The route with the largest total distance is removed (*largest distance-route removal*).
- The route with largest average distance (between two succeeding locations within a route) is removed (*average distance-route removal*). If the average distance of a route is large, i.e. the customer locations are on average relatively far away from each other, the customers might fit better into other routes.

Worst-distance removal

Customers with long distances from their predecessor and successor in the route are removed.

Worst-time removal

Customers are removed considering the time windows and starts of service. On the one hand, long waits should be prevented ($arrival_i \ll RT_i$ with $arrival_i$ being the arrival time at customer location i). On the other hand, those locations where the service starts long after the opening of the time window ($arrival_i \gg RT_i$) are also considered for removal since they might be serviced earlier in the route or in other routes.

Historical knowledge removal

This operator is taking into account historical information. Unlike the neighbour graph removal operator, it does not consider entire solutions but merely the best position found for a customer so far, i.e. the sum of distances to the preceding and succeeding location. Customers with large deviations from their best positions are removed. The best positions are updated whenever new best positions are encountered.

Average distance removal

Customers are removed that deteriorate the average distance of a route. The average distance $\bar{c}_v$ of a route v, which can be described by the sorted customer sequence R_v, is calculated as

$$\bar{c}_v = \frac{\sum_{(i,j)\in R_v} c_{ij}}{n_v}, \tag{4.27}$$

where n_v equals the number of customers in R_v. The cost of a customer i of route v is determined as follows:

$$cost_i = \bar{c}_v - \bar{c}_{v\setminus i}, \tag{4.28}$$

where $\bar{c}_{v\setminus i}$ is the average distance of route v if customer i was not in it ($R_v \setminus \{i\}$). The operator removes customers with high costs.

Node neighbourhood removal

The node neighbourhood removal heuristic initially removes a random customer and then removes customers around it in a rectangular zone.[26] $n_{rem} - 1$ customers are randomly chosen from the customers that are within the zone. If less than $n_{rem} - 1$ customers are in it, the zone is enlarged by a predefined factor.

An example is illustrated in Figure 4.19 where a part of a solution is depicted. In this case, customer 3 is the initially removed customer. The customers 1, 7, 8, 11, and 13 are within the zone and, thus, removed.

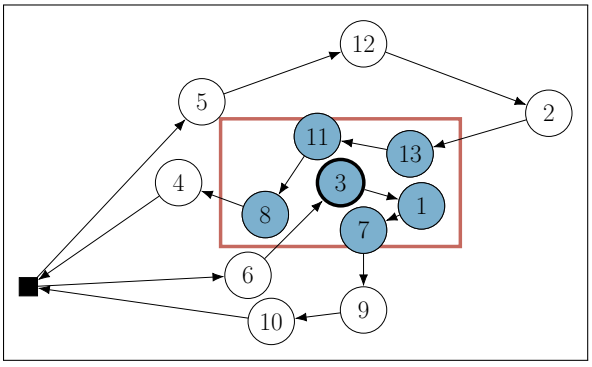


Figure 4.19: Example a for node neighbourhood removal

4.2.2.2 Insertion heuristics

After the application of the removal operator, at least n_{rem} customers are in the set of missing customers MC (so-called *request bank* in Ropke and Pisinger, 2006a,b),

[26] See Demir et al. (2012) for details about the construction of the zone.

i.e., they are not assigned to any route. MC can contain more than n_{rem} customers if more than n_{rem} customers were removed (e.g. by a route removal) or if some customers had already been in the set of missing customers of the current solution. An insertion heuristic is subsequently employed in order to insert the customers from MC into the solution.

The used insertion heuristics are adopted from Ropke and Pisinger (2006a,b). Three basic insertions are used, which are described in the following subchapters. In addition, all three heuristics can be used in combination with a noise factor so that not necessarily the best insertion is realized. That way, the diversification of the search should be strengthened.

Greedy insertion

The first heuristic is a greedy insertion heuristic. In every iteration of the procedure, the customer is chosen for insertion whose best (least cost) insertion into s_{next} is associated with the least cost among all unplanned customers. The basic procedure is outlined in Algorithm 4.27.[27]

Algorithm 4.27 ALNS: greedy insertion heuristic

Input: s_{next}
Output: s_{next}

```
 1: procedure GreedyInsertion
 2:   MC := set of missing customers in s_next
 3:   while |MC| > 0 do
 4:     for each customer j ∈ MC do
 5:       determine best feasible insertion position p*_j
 6:       if j cannot be inserted into any route then
 7:         MC := MC \ {j}
 8:       end if
 9:     end for
10:     i_ins := argmin_{j∈MC} cost(p*_j)          ▷ customer to be inserted
11:     update s_next: insert i_ins into his least-cost position p*_{i_ins}
12:     MC := MC \ {i_ins}
13:   end while
14: end procedure
```

At the beginning of an iteration, the least-cost position p_j^* of each unplanned customer $j \in MC$ is determined (lines 4 to 9). Let, j be a customer to be inserted into a position p between the customers i and k. The cost of the insertion into p is

[27] This pseudocode simply illustrates the general procedure of the heuristic. The actual implementation of the heuristic is described below.

determined as:

$$cost(p) = c_{ij} + c_{jk} - c_{ik}. \tag{4.29}$$

The insertion of j into p_j^* must be feasible regarding the routing and packing constraints listed in Chapter 2.

Regret-k insertion

The regret-k insertions ($k \geq 2$) aim at being more forward-looking than the myopic greedy heuristic. More precisely, they do not only take into account the *best* insertion position for an unplanned customer, but the k best insertion positions. For each customer $j \in MC$ the best insertion position *per route* is determined. Let $p_{1,j}, p_{2,j}, ..., p_{k,j}$ be the first, second, ..., k-th best insertion of j, where each $p_{i,j}$ is the best insertion position of a different route. Their costs are determined as in (4.29). The regret value of j is calculated as:

$$regret_j = \sum_{i=2}^{k} \left(cost(p_{i,j}) - cost(p_{1,j}) \right). \tag{4.30}$$

For example, if $k = 2$, the regret value represents the difference between the best and the second best insertion option.

Within the insertion procedure, the customer with the highest regret value is chosen in each iteration ($i_{ins} := \text{argmax}_{j \in MC}\, regret_j$) and inserted into s_{next} at the best insertion position of i_{ins}, namely $p_{1,i_{ins}}$. If there are one or more customers that cannot be inserted into at least k different routes, the one that can be inserted into the fewest number of routes (but at least one) is chosen for insertion, breaking ties by the insertion cost for the best position.

In this thesis, the regret-2 and regret-3 insertion heuristics are applied. A pseudocode is provided in Chapter 4.2.2.3.

Insertion heuristics with noise factor

Taking into account a noise factor for the calculation of the insertion costs, some randomness is introduced into the choice of the insertion customers. The modified insertion cost $cost'$ for a position p is calculated as

$$cost'(p) = \max(0, cost(p) + noise). \tag{4.31}$$

noise is randomly chosen from the interval $[-\eta \cdot c_{max}, \eta \cdot c_{max}]$, where η is a predefined noise parameter and c_{max} is the maximum cost of all directed edges ($c_{max} = \max_{(i,j)\in E} c_{ij}$).

Apart from the change in the cost calculation, the insertion heuristics proceed as described above. Each of the three mentioned heuristics is also applied with the noise component. Those heuristics are regarded as independent heuristics, so that in total six different insertion heuristics are available for the ALNS: greedy, regret-2, regret-3, greedy with noise, regret-2 with noise and regret-3 with noise.

4.2.2.3 Integration of the packing procedure

In the following, a distinction is made between *packing heuristic* and *packing procedure*. The former refers to the individual heuristics presented in Chapter 4.1. In contrast, the *packing procedure* refers to the packing process in general and describes the procedure, that is called in order to evaluate the feasibility of a route with respect to the packing constraints. Any packing heuristic can be applied within the packing procedure.

Thus, the packing procedure is applied in order to test whether feasible packing plans can be found for generated routes. Depending on the considered 3L-VRPBTW variant and the composition of a route, up to two packing patterns must be generated per route. In addition, the packing procedure checks whether simultaneously transported linehaul and backhaul items overlap during the execution of a route if the side loading approach is applied. The packing part of the hybrid algorithm is usually computationally very expensive: more than 95 % of the computation time of the hybrid routing and packing solution approach is needed for packing. Therefore, calling the packing procedure as rarely as possible is desired.

The packing procedure is required at two points of the hybrid algorithm: within the insertion procedure and when a new best solution is identified. The latter is necessary as routes are not tested for packing feasibility in the course of the removal operators. Removing a customer from a solution can result in a route that cannot be packed feasibly. Thus, infeasible solutions may occur if no further customer is inserted into such a route. In order to ensure the feasibility of the best found solution, the packing procedure is always applied to a new globally best solution and the solution is only accepted if it is feasible. Compliance of all other non-packing

constraints is ensured within the generation of each individual solution.

In the following, the implementation of the insertion heuristics is explained in greater detail. A general framework is presented in Algorithm 4.28. It can be applied to both the greedy heuristic (input parameter $k = 1$) and any regret-k heuristic ($k > 1$).

Algorithm 4.28 ALNS: detailed insertion procedure

Input: s_{next}, k
Output: s_{next}

1: **procedure** INSERTION
2: $MC :=$ set of missing customers in s_{next}
3: $U :=$ set of used routes in s_{next} and one empty route if not all vehicles are used
4: $ip :=$ FALSE ▷ TRUE: at least one insertion is possible
5: **for each** $j \in MC$ **do**
6: $Ins_j := \emptyset$ ▷ set of best feasible insertions of customer j
7: **for each** $v \in U$ **do**
8: $\{ins_{jv}\} :=$ SELECTINS(j, v, s_{next}) ▷ select the best feas. insertion into v
9: $Ins_j := Ins_j \cup \{ins_{jv}\}$
10: **end for**
11: **if** $|Ins_j| > 0$ **then** ▷ at least one insertion is possible for customer j
12: sort Ins_j by non-decreasing insertion costs
13: $ip :=$ TRUE
14: **end if**
15: **end for**
16: **while** $|MC| > 0$ **and** $ip =$ TRUE **do**
17: **if** $k = 1$ **then** ▷ greedy
18: $i_{ins} := \text{argmin}_{j \in MC}\, cost(Ins_j(1))$ ▷ $Ins_j(1)$: cheapest insertion in Ins_j
19: **else** ▷ regret-k
20: **for each** $j \in MC$ **do**
21: $regret_j := \sum_{l=2}^{k}(cost(Ins_j(l)) - cost(Ins_j(1)))$ ▷ calculate regret value
22: **end for**
23: $i_{ins} := \text{argmax}_{j \in MC}\, regret_j$
24: **end if**
25: insert i_{ins} at its minimum cost position into s_{next}
26: $v :=$ route into which i_{ins} was inserted
27: $MC := MC \setminus \{i_{ins}\}$
28: $ip :=$ FALSE
29: **for each** $j \in MC$ **do** ▷ update the best feasible insertion for route v
30: $Ins_j := Ins_j \setminus \{ins_{jv}\}$
31: $\{ins_{jv}\} :=$ SELECTINS(j, v, s_{next})
32: $Ins_j := Ins_j \cup \{ins_{jv}\}$
33: **if** $|Ins_j| > 0$ **then**
34: sort Ins_j by non-decreasing insertion costs
35: $ip :=$ TRUE
36: **end if**
37: **end for**
38: **end while**
39: **end procedure**

The set MC contains all missing, i.e. unassigned, customers. Initially, the best insertion per route is determined for each unassigned customer $j \in MC$ (lines 5 to 15). In this context, the used routes are considered as well as an additional empty route if not all v_{max} available vehicles are used (line 3). The best insertion per route is determined by means of the procedure SelectIns, which is described in greater detail below.[28] Within SelectIns, the packing procedure is applied. The possible insertions of a customer are then sorted by non-decreasing insertion costs. Subsequently, the *actual* insertion procedure starts (lines 16 to 38). In each iteration, i.e. as long as there are unassigned customers left that can be inserted into the solution, the customer with the lowest insertion cost (greedy heuristic) or highest regret value (regret-k heuristic), respectively, is inserted into the solution.[29] After a customer was inserted into a route v, the best insertions of the remaining unassigned customers into route v are updated (line 31). The best insertions into the other routes remain unchanged. Note that unlike in the general framework depicted before (Algorithm 4.27), a customer that cannot be inserted into any route is not excluded from MC since it may be possible to insert that customer into a route after another customer was inserted. As mentioned before, a route that could be packed feasibly can become infeasible when a customer is removed from it. Likewise, it may happen that a customer can be inserted into a route with an additional customer. If the ALNS is applied to a one-dimensional VRP, though, this aspect does not need to be considered and the procedure depicted in Algorithm 4.27 can be applied.

The procedure SelectIns(i, v, s), which returns the best feasible insertion of a given customer i into route v of solution s, is depicted in Algorithm 4.29. Let d_v^L (d_v^B) be the total weight of all linehaul (backhaul) items in route v, and d_i^L (d_i^B) be the total weight of all linehaul (backhaul) items of customer i. Analogously, $vol_v^L, vol_v^B, vol_i^L$ and vol_i^B refer to the corresponding volumes. If the insertion of customer i into route v causes a violation of the weight constraints, no feasible insertion can be returned (line 5). In this regard, only the total weights of the linehaul (backhaul) items loaded at the beginning (end) of the route are considered. The development of the weight inside the vehicle during the route cannot be considered until an insertion position for i is determined. Analogously, the total volumes of the transported and

[28] The best insertion is presented here as a set $\{ins_{jt}\}$ because it may happen that a customer cannot be inserted into a route. In this case, an empty set is returned by SelectIns.

[29] The case that a customer could not be inserted into at least k routes (see above) is omitted here for the sake of simplicity.

potentially inserted items are compared to the vehicle volume capacity.

Algorithm 4.29 ALNS: selection of the best insertion of a customer into a route

Input: customer i, route v, solution s
Output: best insertion Ins^*

```
 1: procedure SelectIns
 2:   Ins := ∅                                   ▷ set of all insertions into route v
 3:   Ins* := ∅                  ▷ set that is returned either with the best insertion
                                   or empty if no insertion is possible
 4:   if d_v^L + d_i^L ≤ D and d_v^B + d_i^B ≤ D and
 5:   vol_v^L + vol_i^L ≤ L · W · H and vol_v^B + vol_i^B ≤ L · W · H then
 6:     for p = 0 to n_v do
 7:       cost(ins_ivp) := cost of inserting i into route v at position p
 8:       Ins := Ins ∪ {ins_ivp}
 9:     end for
10:     sort Ins by non-decreasing cost
11:     while |Ins| > 0 and Ins* = ∅ do
12:       ins := Ins(1)                         ▷ element of Ins with the least cost
13:       R := R_v with i inserted at position p(ins)          ▷ R_v: route v of s
14:       if R is feasible w.r.t. time windows and capacities at each stop then
15:         if R can be packed feasibly then          ▷ call of packing procedure
16:           Ins* := {ins}
17:         end if
18:       end if
19:       Ins := Ins \ {ins}
20:     end while
21:   end if
22: end procedure
```

In a next step (lines 6 to 9), the cost of all potential insertions, i.e. for all positions in the route, are calculated. The costs are determined as mentioned above (4.29). They are then sorted in non-decreasing order and tested successively until a feasible insertion is found or all have been tested. As they are computationally less expensive, feasibility concerning the time window and (weight and volume) capacity constraints are examined first (line 14). Subsequently, the potential new route is tested for packing feasibility (line 15). Hence, in the best case, only one insertion per route must be examined in detail. In the worst case, all potential insertions need to be examined.

Furthermore, a cache is employed as an additional measure for decreasing the packing effort within the hybrid algorithm. It is implemented as a two-dimensional matrix with n rows and up to cs columns where cs represents the maximum cache size. n represents the number of customers in an instance and cs is a predefined parameter. Routes beginning with customer i (after the depot) are stored in row i

of the cache matrix. Before trying to generate a feasible packing plan, the cache, i.e. the row with the respective starting customer, is searched. If the route is not stored there, the packing procedure is applied. In order to facilitate the search within the cache, the routes (represented as vectors of integers, i.e. customer locations) are converted into hash values by means of a hash function. The size of the cache is limited because at some point the search within the cache takes longer than the application of the packing procedure. If a route is to be added to a row of the cache that already contains *cs* elements, the oldest element in the row is removed. Moreover, routes that could be packed feasibly and that could *not* be packed feasibly are stored in the cache with the respective information about their feasibility if a construction packing heuristic is applied (DBLF or TA). If a LS-based heuristic is applied, only routes for which a feasible packing plan could be found are stored. Due to the randomness of the local search, it may happen that a route that could not be packed earlier, can be packed feasibly in another iteration. In order to illustrate the structure of the cache, an example is depicted in Figure 4.20 with the route vectors (which are not stored but presented here for illustration purposes), the corresponding hash values and information about packing feasibility of some routes in the cache.

row \ col.	1	2	3	...	*cs*
1	$(1,2,3)$ 4071742284 $feas =$ TRUE	$(1,6,73,20,51)$ 1089482334 $feas =$ TRUE	$(1,19,3,91)$ 4119203276 $feas =$ TRUE		$(1,36,21,25)$ 129070600 $feas =$ TRUE
2	$(2,42)$ 2838511570 $feas =$ TRUE	$(2,5,92)$ 350135375 $feas =$ FALSE	$(2,86,99)$ 3338004356 $feas =$ TRUE	⋮	
3	$(3,38,2,14,5)$ 1069202073 $feas =$ FALSE	$(3,35)$ 3986927443 $feas =$ FALSE	$(3,21,20,5,98)$ 2677519 $feas =$ TRUE		$(3,65,77,15)$ 491672567 $feas =$ FALSE
⋮		⋱		⋱	...
n					

Figure 4.20: Structure of an exemplary cache

4.2.2.4 Acceptance

Evaluating whether a solution is accepted or not is based on the total cost of a solution, which consists of the sum of the costs of all routes and – if necessary – penalty values. A solution is penalized if more than v_{max} vehicles are used and/or if at least one customer is missing, i.e. not assigned to any route. The usage of more

than v_{max} vehicles can occur due to the savings heuristic, which does not contain any mean for controlling the tour number restriction. Customers can be missing in a solution if they could not be inserted into any route in the insertion process. If less than v_{max} vehicles are used, further tours can be added to a solution in order to include a customer that cannot be inserted into any route. However, if all v_{max} (or more vehicles) are used, no further tour is added and the customer is added to the set of missing customers. By means of the penalty terms, the search is directed towards feasible solutions.

The total cost z' of a solution s is:

$$z'(s) = z(s) + pen_v \cdot \max(0, v_{used} - v_{max}) + pen_{mc} \cdot n_{mc}, \tag{4.32}$$

where v_{used} is the number of vehicles used in s, n_{mc} is the number of missing customers, and z is the sum of the travel distances of all v_{used} routes. Furthermore, pen_v and pen_{mc} are given parameters that penalize the excess number of vehicles and the number of missing customers, respectively. The adherence to all other constraints is ensured within the algorithm.

The acceptance check is embedded into a simulated annealing (SA) framework. In order to leave local optima, the SA approach allows for the deterioration of the current solution in the course of the search. A cooling process represents the reduction of the probability of accepting a worse solution (for further details see, e.g., Talbi, 2009).

After the generation of the initial solution s_{init} for the ALNS, the starting temperature is determined in a way that a solution s with a total cost that is $\theta \cdot 100$ % worse than the cost of s_{init} $(z'(s) = (1 + \theta) \cdot z'(s_{init}))$ is accepted with a probability of 50 %. θ $(\theta > 0)$ is a predefined starting temperature control parameter. That is, initially, the current temperature $Temp$ $(Temp > 0)$ is set to:

$$Temp = \frac{-\theta \cdot z'(s_{init})}{\ln(0.5)}. \tag{4.33}$$

The temperature is "cooling down" in the course of the search causing the probability of accepting bad solutions to decrease. A cooling rate κ $(0 < \kappa < 1)$ is predefined that is applied to the current temperature at the end of each iteration: $Temp := Temp \cdot \kappa$ (Ropke and Pisinger, 2006b).

A solution is always accepted if it is at least as good as the current solution from which it originates ($z'(s_{next}) \leq z'(s_{curr})$). If it is worse than the current solution ($z'(s_{next}) > z'(s_{curr})$), it is accepted to be the new current solution with a probability of $exp\left(\frac{-(z'(s_{next})-z'(s_{curr}))}{Temp}\right)$.

Moreover, if a solution is not accepted due to the above mentioned criteria, it is accepted anyway if it is feasible (with respect to the number of vehicles and missing customers) *and* if the best solution found so far is not feasible. As a feasible solution is usually also connected with lower total costs since it does not contain any penalty costs, it is normally accepted due to being better than a current infeasible solution. It might happen very rarely, though, that an infeasible solution is assigned with lower costs – including penalties – than a feasible one. (This can, in particular, be caused by the choice of penalty terms.) In this case, it should be ensured that the algorithm can return a feasible solution. Therefore, an infeasible solution can only become a new best solution s_{best} if the current s_{best} is also infeasible. On the other hand, s_{curr} can switch between feasible and infeasible solutions throughout the search.

4.2.2.5 Heuristic selection and weight adjustment

In each iteration, a removal heuristic REM and an insertion heuristic INS are applied to the current solution. The respective heuristics are selected based on a roulette wheel selection principle. Each heuristic ψ is assigned with a weight Ω_ψ. Thus, the selection probability of a heuristic depends on its own weight in relation to the weights of the other heuristics. The procedure is implemented as depicted in Algorithm 4.30.

Algorithm 4.30 ALNS: roulette wheel selection

Input: ordered set of heuristics Ψ, weights $\Omega_\psi \; \forall \; \psi \in \Psi$
Output: selected heuristic

```
1:  procedure RouletteWheelSelection
2:      sow := Σ_{ψ∈Ψ} Ω_ψ                          ▷ sow: sum of weights
3:      r := random number from the interval [0, sow]
4:      for ψ := 1 to |Ψ| do
5:          r := r − Ω_ψ
6:          if r ≤ 0 then
7:              return ψ
8:          end if
9:      end for
10: end procedure
```

A predefined parameter *seg* denotes the length of a segment (number of iterations). At the end of a segment, i.e. after *seg* iterations, the weights are updated based on the performances of the heuristics in the previous segment. The performances are recorded in connection with the acceptance checks. For that purpose, each heuristic ψ is assigned a score (scr_ψ) and a counter (cnt_ψ). Both are initialized with the value 0 at the beginning of the search and at the beginning of each segment. cnt_ψ is increased by 1 whenever heuristic ψ is applied. As mentioned above, it could happen that some removal heuristics cannot be applied due the structure of the current solution. In this case, the counter is increased although another removal heuristic is subsequently selected.

The scores $score_{\text{REM}}$ and $score_{\text{INS}}$ are increased by

- ϕ_1, if $z(s_{next}) < z(s_{best})$ (the operation (REM,INS) led to a new globally best solution),
- ϕ_2, if $z(s_{next}) < z(s_{curr})$ (the operation (REM,INS) led to a solution that is better than the current solution),
- ϕ_3, if $z(s_{next}) \geq z(s_{curr})$ but the solution has not been accepted before.

ϕ_1, ϕ_2 and ϕ_3 are given parameters.

The weight Ω_ψ of heuristic ψ ($\psi \in \Psi$) in the next segment is determined as follows:

$$\Omega_\psi := \Omega_\psi \cdot (1 - rct) + rct \cdot \frac{scr_\psi}{cnt_\psi}, \tag{4.34}$$

where rct ($0 \leq rct \leq 1$) is a reaction parameter controlling the impact of the performance on the weight in the next segment.

The approach is based on Ropke and Pisinger (2006b), although it differs in some aspects. In Ropke and Pisinger (2006b), ϕ_2 is only added to the score if the new solution improved the current solution *and* has not been accepted before. In contrast, improvements of the current solution are always rewarded here. Another modification is the addition of ϕ_3 for solutions equally good as the current solution.

4.2.2.6 Termination criteria

The following termination criteria are applied to the ALNS:

- total number of iterations, i.e. the ALNS is terminated after $iter_{max}$ iterations,

- number of iterations without improvement, i.e. the ALNS can be stopped before $iter_{max}$ iterations are reached if no further improvement was generated for $iter_{impr}$ iterations after the last improvement of s_{best},
- time limit, i.e. the ALNS is terminated after a given computing time t_{max}.

The latter is implemented as some instance characteristics lead to high computational efforts especially regarding the packing procedures.

In conclusion, the presented ALNS is based on the works of Ropke and Pisinger (2006a,b). The following aspects of the procedure have been modified and/or extended:

- a savings heuristic is applied in order to determine an initial solution,
- a larger number of removal heuristics is implemented,
- new removal heuristics are proposed,
- the Shaw removal is adapted to the VRPBTW,
- the weight adjustment procedure is modified.

Chapter 5

Numerical experiments

The ALNS, the different packing heuristics and the hybrid approach are evaluated based on extensive numerical experiments. The set-up of these experiments is outlined in Chapter 5.1. Subsequently, the employed instances (Chapter 5.2), the parameter settings as well as the procedure for determining these values are described (Chapter 5.3). Chapter 5.4 contains the findings of the experiments on VRPBTW instances. In Chapter 5.5, the results and analyses regarding 3L-VRPBTW instances are presented including the evaluating of packing heuristics and of hybrid routing and packing approaches.

5.1 Set-up of numerical experiments

The conducted experiments should serve several purposes. First, the best configurations of the ALNS and the hybrid algorithm are determined. Afterwards, the performance of the ALNS and of the hybrid algorithm are assessed. In the course of this, the influence of different instance and problem characteristics, and constraint sets is analysed. In the following, the set-up of the experiments is described shortly. In Chapter 5.2, the instances used for the experiments are presented. For the tuning and the evaluation of the ALNS, 201 popular benchmark instances of various VRPBTW variants are employed. They are presented in detail in Chapter 5.2.1. Furthermore, many instances have been newly generated for the seven considered problem variants (3L-VRPTW, 3L-VRPCB, 3L-VRPCBTW, 3L-VRPMB, 3L-VRPMBTW, 3L-VRPSDP, 3L-VRPSDPTW) and the respective extended problem variants. The instances are generated in a three-stage process. They are used for evaluating the packing heuristics and the hybrid algorithm and are described in Chapter 5.2.2. Finally, CLP instances are used for comparing the packing heuristics to a state-of-the-art CLP algorithm. They are presented shortly in Chapter 5.2.3.
The tuning of the ALNS contains the determination of the most suitable set of parameter values and removal heuristics (Chapter 5.3). A set of 50 test instances of different VRPBTW problem variants is used for this purpose.
Subsequently, the performance of the ALNS is evaluated using all 201 VRPBTW in-

stances. The results of these experiments are presented and analysed in Chapter 5.4. The obtained TTDs are compared to the best known solutions and (if available) to the TTDs obtained by the ALNS of Ropke and Pisinger (2006a).

Chapter 5.5 contains the presentations and analyses of the experiments with the 3L-VRPBTW and CLP instances. First, the best packing construction heuristic of those described in Chapter 4.1 is determined by applying them to randomly generated routes and integrating them into a hybrid savings heuristic (Chapter 5.5.1). Three alternative packing heuristics are employed in the further experiments: the best construction heuristic, the best construction heuristic integrated into the LS framework and the LS with the open space heuristic of Zhang et al. (2015).

Following this, the hybrid algorithms combining the ALNS and the three (alternative) packing heuristics are applied to the 3L-VRPBTW instances (Chapter 5.5.2). These experiments serve to compare the variants of the hybrid algorithm and the different loading approaches. Furthermore, the impact of different instance characteristics (e.g. item size) and problem characteristics (e.g. time windows) is analysed. In the course of these experiments, the constraint set C1 is applied. The constraint sets C2-C5 are applied in additional experiments (Chapter 5.5.3). As before, packing heuristics (considering these constraint sets) are applied to randomly generated routes, and the hybrid savings heuristic and the hybrid ALNS heuristics are employed for solving 3L-VRPBTW instances. The differences in the performance (with respect to solution quality and computing time) between the application of C1 and the sets C2-C5 are analysed.

The algorithms are implemented in C++. The testing of the packing heuristics (Chapter 5.5.1) were performed on a 3.07 GHz and 4 GB RAM computer and the remaining computations with the ALNS and the hybrid algorithm were executed on a Haswell system with Xeon processors with up to 3.2 GHz and 16 GB RAM per core.[30]

5.2 Instances

5.2.1 VRPBTW instances

201 well-known benchmark instances are used in order to compare the results of the ALNS to those from the literature and to evaluate the solution approach. An

[30] See `https://wasd.urz.uni-magdeburg.de/jschulen/urz_hpc/t100/` for more details.

overview is given in Table 5.1.

Table 5.1: Testing instances for the VRPBTW

reference	**abbrev.**	**problem**	**characteristics**	**#**
Solomon (1987)	Sol87	VRPTW	n: 100	56
Goetschalckx and Jacobs-Blecha (1989)	GJB89	VRPCB	n: 25-150; %LH: 80, 67, 50	62
Toth and Vigo (1996)	TV96	VRPCB	n: 21-100; %LH: 80, 67, 50	33
Gélinas et al. (1995)	GDDS95	VRPCBTW	n: 100; %LH: 90, 70, 50	15
Salhi and Nagy (1999)	SN99a	VRPMB	n: 50-199; %LH: 90, 75, 50	21
Salhi and Nagy (1999)	SN99b	VRPSDP	n: 50-199; %LH: see below	14

= number of instances, %LH = share of linehaul customers (in %), abbrev. = abbreviation, n = number of customers per instance

The instances introduced by Solomon (1987) (Sol87) are used for the VRPTW. 56 instances are available with 100 customers each and either wide or narrow time windows. The customer locations are arranged randomly, clustered or mixed (partly clustered, partly random). In the literature regarding the VRPTW, the (first) optimization objective is often the minimization of the number of used vehicles (see Chapter 3.1.3). As the optimization criterion employed in this thesis is the minimization of the TTD, the known minimum TTD solutions of the Sol87 instances are used as benchmarks. Considering this criterion, optimality has been proven for 55 out of the 56 instances.

One set of instances used for the VRPCB, is the set proposed by Goetschalckx and Jacobs-Blecha (1989) (GJB89). It consists of 15 test problems with 25 to 150 randomly allocated customer locations, which include 80, 67 or 50 % linehaul customers. For each problem, the number of vehicles and the vehicle capacity is varied so that three to six variants are generated for each problem. All in all, 62 instances are available.

A second set of VRPCB instances is presented by Toth and Vigo (1996) (TV96) consisting of 33 instances that are based on well-known CVRP instances and include 80, 67 or 50 % linehaul customers. VRPCB instances are generated by converting every fifth, third or second customer, respectively, of the CVRP instance into a backhaul customer. Optimal solutions for some instances are provided, for example, by Toth and Vigo (1997). However, in their work and also in some others it is assumed that all vehicles *must* be used. Hence, these solutions are not necessarily

optimal for the problem investigated here where a limited number of vehicles *can* be used. The results of the ALNS presented in this thesis are exclusively compared to other solutions where a maximum available number of vehicles is considered.

Furthermore, the VRPCBTW instances of Gélinas et al. (1995) (GDDS95) are utilized. They are based on five instances of Solomon (1987) with randomly allocated customer locations and narrow time windows. 10, 30 or 50 % of the customers are randomly chosen to be backhaul customers. Thus, 15 instances are available.

Instances of Salhi and Nagy (1999) are used for the VRPMB (SN99a) and the VRPSDP (SN99b). They are based on seven popular CVRP instances. The VRPMB instances from set SN99a are created by converting every second, fourth or tenth customer into a backhaul customer resulting in 21 VRPMB instances.[31]

In order to obtain VRPSDP instances, the seven CVRP instances mentioned above are used and Salhi and Nagy (1999) split each customer's demand. Let, d_i be the demand of customer i, and (cx_i, cy_i) be the coordinates of the location of i. A ratio r_i is determined for each customer with

$$r_i = \min\left(\frac{cx_i}{cy_i}, \frac{cy_i}{cx_i}\right). \tag{5.1}$$

The linehaul demand of a customer i equals $r_i \cdot d_i$ and the backhaul demand equals $(1 - r_i) \cdot d_i$. Further seven instances are generated by exchanging the pickup and demand quantities for every second customer. Consequently, 14 instances are available in the set SN99b.

In the following, the described instances are referred to as (one-dimensional) VRPBTW instances. Analogously, the corresponding optimization problems are referred to as (one-dimensional) VRPBTWs.

5.2.2 3L-VRPBTW instances

Since there are no benchmark instances available that cover all of the aspects of the problems considered here – i.e. three-dimensional loading constraints, time windows, backhauls – new instances are generated.[32] In doing so, some traits of well-known benchmark instances are adopted.

[31] Additional 21 instances contain drop times and maximum distance constraints, but regarding the customer coordinates and demands they are duplicates of the former instances. These additional instance are omitted here since the maximum distance constraint is not considered.

[32] The 3L-VRPBTW instances of Reil et al. (2017) were published too late to be used for this thesis.

Obtaining the instances for the extended 3L-VRPBTW variants is a three-stage process. The aim of this process is to obtain instances that differ significantly with regard to various characteristics. Yet, they are to be generated in a way that the instances of different extended problem variants have certain characteristics in common. That way, analyses are possible that serve to compare the problem features and their influence on the solutions and on the performance of the hybrid algorithm. In the course of this process, different instance types are distinguished, which are explained in greater detail below. Each instance type corresponds to a stage of the generation process. One of these types are so-called basic instances. The generation of the characteristics of a basic instance is described in Chapter 5.2.2.1. In Chapter 5.2.2.2, the connection of the basic instances to the remaining instance types is presented.

5.2.2.1 Generation of basic instances

A basic instance is defined by the following traits: the number of customer locations and their distribution, time windows, the number of item types and their characteristics, the allocation of items to the customers, the separation of customers and items into linehauls and backhauls and the loading space dimensions and capacities of the vehicles.

Number and distribution of customers

Instances are generated with 20, 60 and 100 customers. Their distribution is based on (a subset of) the instances of Solomon (1987) with a random allocation of the customer locations. The depot is placed at a central position at the coordinates (35, 35) and the customers are randomly (uniformly) distributed in a Cartesian coordinate system from 0 to 70 on the abscissa and 0 to 80 on the ordinate. The coordinates from the Sol87 set with randomly allocated customer locations is used for one basic instance for the 3L-VRPBTW. The coordinates for the remaining basic instances are generated independently.

Generation of time windows

The structures of the time windows resemble those of the instances of Solomon (1987) and basic instances with either wide or narrow time windows are created. The depot ready time is set to $RT_0 = 0$ for both narrow and wide time windows,

and the due date to $DD_0 = 230$ for basic instances with narrow time windows and $DD_0 = 1000$ for basic instances with wide time windows. Not necessarily all customers are assigned a time window. Each basic instance has a different time window density defining the proportion of customers with time windows. The time window density is randomly chosen from the set $\{25\ \%, 50\ \%, 75\ \%, 100\ \%\}$. The respective customers are determined randomly. If a customer i does not have a time window, a fictional time window $(0, DD_0 - c_{i0} - ST_i)$ is created, where ST_i is the service time at customer location i and c_{i0} is the required travel time between i and the depot. For each basic instance, a service time of 10 time units is assumed for each customer location and of 0 time units for the depot. If a customer i is assigned a time window, the centre of the time window is chosen randomly and uniformly distributed in the interval $(RT_0 + c_{0i}, DD_0 - c_{i0} - ST_i)$. The time window width is generated as a normally distributed random number with the mean μ and standard deviation σ listed below (Table 5.2). The values for the parameters μ and σ are adopted from Homberger (2000).

Generation of item types and allocation of items

As opposed to the instances of Gendreau et al. (2006), no individual items are generated but different item types in order to assess the impact of various degrees of heterogeneity. Basic instances are generated with three, ten or 100 different item types. The dimensions of the item types are generated similarly to the procedure described by Gendreau et al. (2006).

In the following, let the lengths, weights and volumes be given in length units (LU), weight units (WU), and volume units (VU), respectively. Gendreau et al. (2006) generate the length of an item uniformly randomly distributed in the interval $[0.2L, 0.6L]$. Accordingly, the width and height are chosen from the respective intervals $[0.2W, 0.6W]$ and $[0.2H, 0.6H]$. However, since for the 3L-VRPMB(TW) and 3L-VRPSDP(TW) the loading approach with the separated loading spaces of height $0.5H$ is applied, the items must not be higher than $0.5H$. Thus, the original intervals for the length and width are adopted from Gendreau et al. (2006), i.e. the length l_τ of an item type τ is uniformly randomly chosen from the interval $[0.2L, 0.6L]$ and its width w_τ from the interval $[0.2W, 0.6W]$. The height h_τ is randomly chosen from $[0.2H, 0.5H]$. The respective items are hereinafter referred to as large items. Their volumes amount to 0.8 % to 18 % of the loading space volume.

Furthermore, it shall be tested how the performance of the packing heuristics and the hybrid algorithm is affected by the item size. Therefore, basic instances with small item types are additionally generated. These item types are randomly assigned a length (width, height) of $[0.1L, 0.3L]$ ($[0.1W, 0.3W]$, $[0.1H, 0.3H]$). Thus, their volumes amount to 0.1 % to 2.7 % of the loading space volume.

Let $vol_\tau = l_\tau \cdot w_\tau \cdot h_\tau$ be the volume of item type τ. The weight d_τ of item type τ is determined randomly and depending on the item volume. It is a uniformly distributed value from the interval $\left[0.001 \left\lceil \frac{WU}{VU} \right\rceil \cdot vol_\tau[VU], 0.01 \left\lceil \frac{WU}{VU} \right\rceil \cdot vol_\tau[VU]\right]$.

20 %, but at least one, of the item types are determined to be fragile. The load bearing strength of an item type is generated in the following way: Initially, the minimum and maximum weight per area unit (wpa_{min}, wpa_{max}) of all item types is determined ($wpa_{min} = \min_{\tau \in T} \frac{d_\tau}{l_\tau \cdot w_\tau}$, where T is the set of all item types; wpa_{max} analogously). The minimum value serves to ensure that each item type can be carried by at least one item type (even if it is the respective type itself). The load bearing strength p_τ of an item type τ is then determined as $p_\tau = r \cdot wpa_{max}$, where r is a continuous random number that is uniformly randomly chosen from the interval $[1, 4]$ if $f_\tau = 0$ (not fragile) and $[0.5, 1]$ if $f_\tau = 1$ (fragile). The load bearing strength of each item type must be greater than or equal to wpa_{min}.

The total number of items m is a fixed instance parameter. Instances are generated with either 200 or 400 items. Each customer is assigned a number of items selected randomly from a predefined interval in a way that the total number of demanded items adds up to the given total number of items. The intervals are as follows:

- 5 to 15 items per customer for $n = 20$ and $m = 200$,
- 10 to 30 items per customer for $n = 20$ and $m = 400$,
- 2 to 5 items per customer for $n = 60$ and $m = 200$,
- 3 to 10 items per customer for $n = 60$ and $m = 400$,
- 1 to 3 items per customer for $n = 100$ and $m = 200$,
- 2 to 6 items per customer for $n = 100$ and $m = 400$.

Each item of a customer is randomly assigned to an item type and it adopts the specifications of the respective type.

Linehaul shares

Depending on the problem variant, either the customers or the items must be subdivided into linehaul and backhaul customers or items, respectively. As a basic instance can be transferred to instances for all problem variants, it contains the division of both customers *and* items into linehauls and backhauls. The linehaul share is a prescribed parameter. Basic instances with linehaul shares of 50 % and 80 % are generated. That is, the first 50 % or 80 %, respectively, of the customers are determined to be linehaul customers. The remaining customers are backhaul customers. Moreover, 50 % or 80 % of the items of each customer are randomly determined to be linehaul items. The remaining items are backhaul items.

Generation of the vehicle fleet

The dimensions of the loading space are set to $L = 60\,LU, W = 25\,LU$ and $H = 30\,LU$ and the weight capacity to $D = 200\,WU$ for all instances. A vertical stability parameter of $VSP = 0.75$ and a maximum reach of 5 LU is assumed.

In order to guarantee that at least one feasible solution exists for each instance, it is ensured that all items of every single customer fit into the loading space by applying a packing construction heuristic.

Instance classes

The described procedure aims at acquiring classes of basic instances that differ noticeably from each other. A basic instance class is defined by the number of customers, the total number of items, the item size, the number of item types, the linehaul share and the time window width. The settings of these features are presented in Table 5.2.

No basic instances are generated for $n = 20$ and large items. Due to the relatively large number of items per customer, the items of one customer alone would fill up almost the whole loading space. Hence, no sensible routing problem can be constructed. Moreover, in the case of $m = 400$, every customer is assigned on average 20 items. However, these items would not fit into the loading space unless they would be rather small, i.e. all dimensions would be close to 0.2 times the respective dimension of the loading space. This would defeat the purpose of investigating large items.

Combining the six instance characteristics and settings (with the mentioned ex-

ception), 120 different instance classes are obtained. For each instance class, five different basic instances are generated, i.e. 600 in total.

Table 5.2: Instance characteristics

instance parameter	**characteristics and values**
number of customers (n)	20, 60, 100
total number of items (m)	200, 400
item size	large (intervals for length/width/height: $[0.2L, 0.6L]$/ $[0.2W, 0.6W]$/ $[0.2H, 0.5H]$) (not for $n = 20$), small (intervals for length/width/height: $[0.1L, 0.3L]$/ $[0.1W, 0.3W]$/ $[0.1H, 0.3H]$)
number of different item types	3, 10, 100
linehaul share	50 %, 80 %
time windows	wide ($\mu = 240, \sigma = 60$), narrow ($\mu = 60, \sigma = 10$)

5.2.2.2 Instance types

Basic instances can be used to derive problem instances for the specific extended problem variants. As mentioned above, this is a three-stage process. On the first stage, the basic instances are created. Secondly, so-called *core instances* are generated. A core instance contains the necessary information for a problem variant (i.e. backhaul and time window variant). Finally, *instances* contain all necessary problem information in order to be solved, i.e. they correspond to a given extended problem variant.

That is, seven core instances are derived from each basic instance. The number and distribution of the customers, the item properties and their allocation to the customers and the vehicle characteristics are adopted from the respective basic instance. Furthermore, if a core instance corresponds to a problem variant with time windows, the time windows of the basic instance are adopted. Otherwise, they are disregarded. In addition, the linehaul-backhaul-separation needs to be extracted for the respective problem variant. That is, in a core instance for the 3L-VRPTW, both the separation of the customers and of the items is disregarded (all customers are linehaul customers and demand linehaul items only). Core instances for the 3L-VRPCB(TW) or 3L-VRPMB(TW) adopt the separation of the customers into linehaul and backhaul customers from the basic instance, but disregard the separation of items. Analogously, the separation of items is adopted and the separation of the customers is disregarded if a core instance belongs to the 3L-VRPSDP(TW).

Furthermore, the number of available vehicles is determined on the core instance level. For this purpose, each core instance is additionally provided with a certain loading approach and a constraint set. For the 3L-VRPTW and 3L-VRPCB(TW), the rear loading approach is considered. The LSP approach is taken into account for the 3L-VRPMB(TW) and 3L-VRPSDP(TW). For all problem variants, the constraint set C1 is regarded. The resulting instances are solved using a hybrid savings algorithm in which TA-Walls is employed as the packing heuristic. Pre-tests have shown that this heuristic performs relatively poorly compared to other packing heuristics (see below, Chapter 5.5.1). Therefore, obtaining feasible solutions with TA-Walls should guarantee that better heuristics are also able to generate feasible solutions. The numbers of available vehicles are determined by the numbers of tours formed by this procedure. Core instances with 400 items are neglected for the variants with mixed backhauls (3L-VRPMB, 3L-VRPMBTW) and simultaneous delivery and pickup (3L-VRPSDP, 3L-VRPSDPTW) as these problem variants are more complicated than the other ones (due to the simultaneous transport of linehaul and backhaul items).

In the final step, instances for the extended problem variants are derived from the core instances. The instances adopt the number and distribution of the customers, time windows (if applicable), the item properties and their allocation to the customers, the separation of customers or items (if applicable), and the size and characteristics of the vehicle fleet from the core instances. In addition, the loading approach and constraint set is determined for each instance. Based on the extended problem variants defined above (see Chapter 2.1, Table 2.2), ten instances are derived from each core instance.

An overview of this procedure is presented in Table 5.3.

Table 5.3: Overview of instance types

instance type	**problem variant**	**instance characteristics**									#	Σ
		cust. (no./ distr.)	time windows	items	LH/BH (cust.)	LH/BH (items)	vehicles (dim./ capa)	vehicles (no.)	loading approach	constraint set		
basic instance: for all problem variants	3L-VRPBTW	def. in BI	def. in BI	def. in BI	def. in BI	def. in BI	def. in BI	-	-	-	600	600
core instance: for a problem variant	3L-VRPTW	from BI	from BI	from BI	n.a.	n.a.	from BI	def. in CI	-	-	600	3,000
	3L-VRPCB	from BI	n.a.	from BI	from BI	n.a.	from BI	def. in CI	-	-	600	
	3L-VRPCBTW	from BI	from BI	from BI	from BI	n.a.	from BI	def. in CI	-	-	600	
	3L-VRPMB	from BI	n.a.	from BI	from BI	n.a.	from BI	def. in CI	-	-	300*	
	3L-VRPMBTW	from BI	from BI	from BI	from BI	n.a.	from BI	def. in CI	-	-	300*	
	3L-VRPSDP	from BI	n.a.	from BI	n.a.	from BI	from BI	def. in CI	-	-	300*	
	3L-VRPSDPTW	from BI	from BI	from BI	n.a.	from BI	from BI	def. in CI	-	-	300*	
instance: for an extended problem variant (*examples*)	(3L-VRPTW,RL,C1)	from CI	from CI	from CI	from CI	from CI	from CI	from CI	def. in I	def. in I	600	30,000
	(3L-VRPCB,SL,C2)	from CI	from CI	from CI	from CI	from CI	from CI	from CI	def. in I	def. in I	600	
	(3L-VRPMBTW,SL,C3)	from CI	from CI	from CI	from CI	from CI	from CI	from CI	def. in I	def. in I	300*	
	(3L-VRPSDP,LSP,C5)	from CI	from CI	from CI	from CI	from CI	from CI	from CI	def. in I	def. in I	300*	
	...	...	...	...	...	...	...	...	...	...	...	

-: not defined, #: quantity, *: reduced number of core instances due to neglecting those with 400 items, BI: basic instance, CI: core instance, cust. (no./ distr.): number and distribution of customers, def. in ...: defined in ..., I: instance, LH/BH: linehaul/ backhaul separation (with respect to...), n.a.: not applied, vehicles (dim./ capa.): dimensions and capacity of vehicles, vehicles (no.): number of vehicles, Σ: sum

5.2.3 CLP instances

The CLP instances proposed by Bischoff and Ratcliff (1995) and Davies and Bischoff (1999) are utilized for testing the packing heuristics. These instances consist of 15 sets with 100 instances each. The sets differ by the heterogeneity of the items, i.e. they have different numbers of item types ranging from three to 100. An orientation constraint with respect to height is prescribed, that is, only two or four spatial orientations are permitted for some items of an instance. Furthermore, it is assumed that all items must be fully supported ($VSP = 1$).

5.3 Parameter settings and configurations of the hybrid algorithm

A set of 50 benchmark instances of all VRPBTW variants is used for determining the best parameter values for the ALNS. Based on the settings of Ropke and Pisinger (2006b), several values for each parameter are tested by solving each test instance five times (due to random components) and keeping the remaining parameter values fixed. The only parameters referring to the packing procedures are max_enum and cs. The value proposed by Zhang et al. (2015) for max_enum is adopted. The final settings are listed in Table 5.4.

Table 5.4: Parameter settings for the hybrid solution approach

parameter	**description**	**value**
$iter_{max}$	maximum number of iterations	25,000
$iter_{impr}$	maximum number of iterations without improvement	8,000
t_{max}	time limit [min]	15 for $n = 20$,
		60 for $n \geq 60$
no_{min}, no_{max}	interval for number of removed customers	$0.04n, 0.4n$
ϕ_1, ϕ_2, ϕ_3	weight adjustment parameters	50, 10, 5
$\omega_1, \omega_2, \omega_3, \omega_4$	Shaw weights	6, 3, 2, 6
ρ	determinism parameter	6
rct	reaction factor	0.8
seg	segment length	100
η	noise parameter	0.025
κ	cooling rate	0.99975
θ	starting temperature control parameter	5 %
pen_v	penalty term for violation of the tour number restriction ($c_{max} = \max_{(i,j)\in E} c_{ij}$)	$10 \cdot c_{max}$
pen_{mc}	penalty term for missing customers	$1 \cdot c_{max}$
max_enum	maximum number of items in a route up to which all item sequence permutations are packed	8
cs	cache size	500

In addition, the test instances are used for determining the most suitable set of removal heuristics. Different combinations of the heuristics described in Chapter 4.2.2.1 are tested and the results are compared regarding the obtained average TTDs and the numbers of best solutions found among all tested sets of removal heuristics. The detailed results are provided in Appendix A. The TTDs do not differ noticeably among the various combinations. As a consequence, the set with the most best solutions is selected. It consists of the removal heuristics of Ropke and Pisinger (2006a,b):

- Shaw removal,
- random removal,
- worst removal,
- cluster removal,
- neighbour graph removal,

and some of the newly developed ones:

- overlap removal,
- inner route removal,
- intersection removal,
- route pair removal.

This set achieves one of the lowest values of average deviation from the best known solutions (BKS). Further conclusions of these experiments are that using multiple removal heuristics is beneficial to the ALNS performance. However, the ALNS is also "saturated" with a few heuristics. A very large number of heuristics does not contribute more to the solution quality than a comparatively low number of about ten heuristics. More heuristics neither contribute significantly to the solution quality nor deteriorate it.

Moreover, the weight development of the removal heuristics was recorded in pretests. That way, some heuristics could be identified as allegedly *good* heuristics due to having high weights throughout and also at the end of the search. Therefore, one of the heuristic sets contains only these *good* heuristics (random, worst, neighbourhood graph, historical knowledge, average distance, and intersection removal). Interestingly, this set of removal heuristics leads to one of the worst results – both in terms of average deviations from the BKS and number of best solutions found. One can conclude that the ALNS should be equipped with a well balanced set of removal heuristics, which is able to react to a variety of instance characteristics.

5.4 Results for VRPBTW instances

In order to evaluate the quality of the routing algorithm, 201 instances of different VRPBTW variants are used. Each instance is solved five times by the ALNS. The results are summarized in Table 5.5. It contains the average deviations of the best (ALNS best) and average (ALNS avg) TTDs obtained by the ALNS from the TTDs of the best known solutions (∅dev BKS). In addition, average deviations of the TTDs of the best ALNS solutions from those of the best solutions reported by Ropke and Pisinger (2006a) (∅dev RP06) are stated if available. The number of instance for which new best solutions are obtained by the ALNS are given (new best) as well as the average computing times (∅ct) per instance and run in seconds. Detailed results for all instance sets are provided in Appendix B.
Here and in the following, deviations from certain benchmarks (here: BKS from the literature) are presented. The deviation $dev(p)$ of the TTD obtained by a procedure p from the benchmark BM is determined as:

$$dev(p) = \frac{TTD(p) - TTD(BM)}{TTD(BM)}. \tag{5.2}$$

Thus, negative values imply improvements compared to the benchmark.

Table 5.5: Average deviations of TTDs provided by the ALNS from TTDs of benchmark solutions, numbers of new best solutions and average computing times per instance for VRPBTW instance sets

problem	**set**	**∅dev BKS[%]**		**∅dev RP06[%]**	**new**	**∅ct[s]**
		ALNS best	ALNS avg	ALNS best	**best**	
VRPTW	Sol87	0.50	0.68	–	1	31.90
VRPCB	GJB89	0.08	0.27	−0.37	10	26.38
VRPCB	TV97	0.76	0.97	0.66	1	9.79
VRPCBTW	GDDS95	0.72	0.81	–	6	55.67
VRPMB	SN99a	0.14	0.36	0.07	1	65.08
VRPSDP	SN99b	−0.84	−0.34	−4.40	7	59.85
total		0.30	0.51	−0.47	26	33.75

∅: average, ct: computing time per instance and run, dev BKS: deviation from TTDs of best known solution, dev RP06: deviation from best TTDs of Ropke and Pisinger (2006a), new best: number of instances with new best solutions

The results verify the competitiveness of the ALNS. The TTDs of the best solutions per instance deviate on average only 0.3 % from the TTDs of the BKS. Very good results are obtained in particular for the instance set SN99b with an average deviation of the best solutions from the BKS of −0.84 %. In total, 26 new best

solutions are obtained. As mentioned before, in all cases the optimization criterion is the minimization of the TTD (cf. Chapter 2). For the VRP variants including time windows, the minimization of the number of vehicles is often targeted in the literature. The BKS used as references here are the best solutions regarding the TTD, though. Almost all instances of Solomon (1987) had previously been solved to optimality. Only one was – to the best of the author's knowledge – not solved optimally yet. The ALNS achieved to improve the best known solution for that instance. For the remaining instances, the ALNS solutions are very close to the optimal solutions with an average deviation of the TTDs of the best solutions from the TTDs of the BKS of 0.56 %[33].

The results are generated within reasonable computing times. The overall average computing times amount to about 34 seconds. The computing times depend heavily on the number of customers. Instances with up to 30 customers can be solved on average within less than one second and on average less than 30 seconds are needed for instances with less than 100 customers. The maximum time of about 295 seconds is required by an instance with 199 customers.

Several components of the original procedure have been modified and, thus, contributed to the improved performance of the ALNS: a larger number of removal heuristics, construction of the initial solution by means of the savings heuristic, Shaw removal heuristic adapted to the VRPBTW and the weight adjustment procedure. Moreover, most parameters have been tuned similarly to Ropke and Pisinger (2006a,b). However, the value chosen for the parameter r (reaction factor) is significantly higher (here: $r = 0.8$; Ropke and Pisinger (2006a,b): $r = 0.1$). That is, the algorithm reacts stronger to the performances of the individual heuristics.

5.5 Results for 3L-VRPBTW and CLP instances

In this section, packing heuristics, loading approaches and hybrid solution approaches are evaluated. First, the experiments regarding the packing heuristics are analysed in Chapter 5.5.1. These experiments serve to identify the most suitable packing construction heuristic, which is then used for further experiments and additionally integrated into the local search-framework. Finally, three alternative packing procedures are combined with the ALNS: the construction heuristic selected

[33] This value is not included in the table.

before, the local search (LS) with the selected construction heuristic and the LS with the open space heuristic of Zhang et al. (2015). The results of the hybrid approaches are analysed in Chapter 5.5.2.

5.5.1 Evaluation of packing heuristics

The following packing construction heuristics are evaluated: DBLF, DBLF$^+$, DBLF-Comb, TA-Walls, TA-NoWalls, TA$^+$-Walls and TA$^+$-NoWalls (see Chapter 4.1.2 for details). In addition, the LS-based approach of Zhang et al. (2015) and the LS combined with the best construction heuristic, determined in the course of the experiments, are regarded. First, the heuristics are applied in order to generate packing plans for randomly generated routes. Furthermore, the packing heuristics are integrated into the savings algorithm and the resulting hybrid approaches are utilized in order to solve 3L-VRPBTW instances. The obtained solutions are compared with respect to the TTDs. Based on both experiments, the best packing construction heuristic is selected.

5.5.1.1 Randomly generated routes

More than 35,000 routes have been generated randomly using the instances of the extended problem variant (3L-VRPTW, RL, C1)[34] presented in Chapter 5.2.2. The routes differ in the required volume utilization of the transported items (20-90 % for instances with small items, 10-60 % for instances with large items). Routing constraints (including time window constraints) are not regarded for the generation of the routes. The packing constraint set C1 based on Gendreau et al. (2006) and the rear loading approach are applied. In the following, it is analysed how many routes could be packed feasibly by the packing heuristics. Table 5.6 shows the results of the experiments. The table contains the shares of routes that are packed feasibly, i.e. for which a feasible packing plan is found, by applying the respective construction heuristics. The best results among the heuristics (the highest shares of feasibly packed routes) are bold-faced.

For both large and small item instances, the best results are obtained by the DBLF heuristic (small items: 48.6 % of the routes are feasibly packed, large items: 61.4 %). TA-Walls also achieves good results for large items (57.9 % of the routes are feasibly

[34] See Chapter 2.1.2.5 for the introduction of this notation and the used abbreviations.

packed) and it is even the dominating heuristic for volume utilizations of 20-30 %. However, all of the TA heuristics perform comparatively poorly for small items (TA-Walls/ TA$^+$-Walls: 13.8 %, TA-NoWalls/ TA$^+$-NoWalls: 21.7 %). Interestingly, the variants TA-/TA$^+$-Walls, which take into account the container walls, perform (slightly) better than the variants TA-/TA$^+$-NoWalls in the case of large items. For small items, though, TA-/TA$^+$-NoWalls lead to considerably better results. Furthermore, the extension regarding the placement tests (see Chapter 4.1) does not have any influence on the results of the TA heuristics. For large items, the performance of DBLF$^+$ and DBLF-Comb is only slightly worse than DBLF. For small items, however, they perform considerably worse than DBLF, but in any case better than the TA heuristics.

Table 5.6: Comparison of packing construction heuristics; shares of feasibly packed random routes

heuristic	**share of feasibly packed routes** [%]								
vol [%]	10-20	20-30	30-40	40-50	50-60	60-70	70-80	80-90	total
small items									
DBLF		**99.8**	**97.9**	**78.8**	**48.8**	**26.9**	**10.5**	**0.8**	**48.6**
DBLF$^+$		89.3	73.8	50.1	30.9	19.1	6.9	0.4	35.0
DBLF-Comb		89.3	73.8	50.1	30.9	19.1	6.9	0.4	35.0
TA-Walls		50.2	31.6	18.2	9.9	3.5	1.0	0.1	13.8
TA-NoWalls		72.2	49.6	29.0	16.7	6.8	1.5	0.1	21.7
TA$^+$-Walls		50.2	31.6	18.2	9.9	3.5	1.0	0.1	13.8
TA$^+$-NoWalls		72.2	49.6	29.0	16.7	6.8	1.5	0.1	21.7
large items									
DBLF	**98.6**	88.5	**66.9**	**37.6**	**15.4**				**61.4**
DBLF$^+$	98.0	85.5	61.5	32.5	14.0				58.3
DBLF-Comb	98.0	85.5	61.5	32.5	14.0				58.3
TA-Walls	**98.6**	**88.8**	61.8	28.8	11.3				57.9
TA-NoWalls	97.8	83.3	57.2	28.0	10.3				55.3
TA$^+$-Walls	**98.6**	**88.8**	61.8	28.8	11.3				57.9
TA$^+$-NoWalls	97.8	83.3	57.2	28.0	10.3				55.3

vol: volume utilization of the loading space (utilized by transported items).
Maximum shares per column are bold-faced. Empty cells indicate that no routes are considered for the respective volume utilization interval.

The *plain* DBLF heuristic performs best in the experiments with randomly generated routes. Thus, it is selected for further experiments and integrated into the LS framework. In the following, the latter heuristic will also be referred to as LS_DBLF (and as LS_DBLF^{SL} in the case of the modified variant for the side loading approach). For comparison, the heuristics LS_DBLF and LS_OS are also applied to the randomly generated routes. Their results are presented in Table 5.7 (together with the results

obtained by DBLF as comparison). For all levels of utilization and for both large and small items, the LS_OS heuristic achieves the best results (small items: 60.9 %, large items: 83.9 %) and both LS heuristics clearly outperform the construction heuristic. The differences are greater when considering large items. Here, the differences in the shares of feasibly packed routes between DBLF and LS_DBLF (LS_OS) amount to 16 (22.5) percentage points. In the case of small items, they amount to 10.1 (12.3) percentage points.

Table 5.7: Comparison of packing heuristics (DBLF, LS_DBLF, LS_OS); shares of feasibly packed random routes

heuristic	**share of feasibly packed routes** [%]								
vol [%]	10-20	20-30	30-40	40-50	50-60	60-70	70-80	80-90	total
small items									
DBLF		99.8	97.9	78.8	48.8	26.9	10.5	0.8	48.6
LS_DBLF		99.9	99.9	93.7	68.0	41.1	22.3	3.8	58.7
LS_OS		**100.0**	**100.0**	**98.4**	**74.0**	**43.6**	**23.0**	**4.0**	**60.9**
large items									
DBLF	98.6	88.5	66.9	37.6	15.4				61.4
LS_DBLF	**100.0**	98.5	91.0	64.3	33.2				77.4
LS_OS	**100.0**	**99.6**	**97.6**	**79.8**	**42.3**				**83.9**

vol: volume utilization of the loading space by transported items.
Maximum shares per column are bold-faced. Empty cells indicate that no routes are considered for the respective volume utilization interval.

5.5.1.2 Hybrid savings heuristic

In the following, the results of the experiments of the savings heuristic combined with different (alternative) packing heuristics are presented and examined. For this purpose, abbreviations are utilized in order to refer to the different hybrid approaches. For example, Sav×DBLF refers to the savings heuristic combined with the DBLF heuristic. The remaining approaches are abbreviated analogously.

For these experiments, the seven 3L-VRPBTW variants presented above are considered. The variants without backhauls and with clustered backhauls are only solved using the standard rear loading approach, while the LSP is applied to the problems with mixed backhauls and simultaneous delivery and pickup. That is, the following extended problem variants are considered: (3L-VRPTW/3L-VRPCB(TW), RL, C1), (3L-VRPMB(TW)/3L-VRPSDP(TW), LSP, C1). Each instance of the considered extended problem variants is solved once by each variant of the hybrid savings heuristic.

The results are summarized in Table 5.8. The objective function values obtained by Sav×DBLF for each problem variant serve as the benchmarks here. In Table 5.8, the average deviations from the benchmarks are presented (∅dev TTD BM), over all instances and subdivided by the item sizes. In addition, the average computing times per instance (in seconds) are given (∅ct).

Among the approaches with the construction heuristics, the best solutions are obtained by Sav×DBLF. Regarding the TA heuristics, the algorithm variants including TA$^+$-/TA-Walls outperform the algorithm variants with TA$^+$-/TA-NoWalls for large items and vice versa for small items. The average deviations from the benchmarks amount to 1.79 % (TA$^+$-/TA-Walls) and 2.37 (TA$^+$-/TA-NoWalls) if the items are large. If they are small, they amount to 4.87 % (TA$^+$-/TA-Walls) and 1.83 % (TA$^+$-/TA-NoWalls). Moreover, the best results over all heuristics are achieved by Sav×LS_DBLF (average benchmark deviation of −1.34 %) and Sav×LS_OS (−1.99 %). In particular in the case of large items, the Sav×LS_OS heuristic is clearly dominating.

Table 5.8: Comparison of savings heuristics combined with different packing heuristics; average deviations of benchmark (Sav×DBLF) TTDs and average computing times

	savings heuristic					
	∅dev TTD BM [%]			**∅ct[s]**		
pack	small items	large items	total	small items	large items	total
DBLF	0.00	0.00	0.00	0.03	0.01	0.02
DBLF$^+$	0.56	1.16	0.70	0.06	0.01	0.05
DBLF-Comb	0.56	1.16	0.70	0.03	0.01	0.02
TA-Walls	4.87	1.79	4.15	0.05	0.01	0.04
TA-NoWalls	1.83	2.37	1.95	0.04	0.01	0.04
TA$^+$-Walls	4.87	1.79	4.15	0.10	0.02	0.08
TA$^+$-NoWalls	1.83	2.37	1.95	0.11	0.02	0.09
LS_DBLF	**−0.55**	−3.95	−1.34	0.77	0.04	0.60
LS_OS	**−0.55**	**−6.72**	**−1.99**	2.12	0.18	1.67

∅: average, ct: computing time per instance, dev TTD BM: deviation from benchmark TTDs, pack: integrated packing heuristic.
Minimum average deviations per column are bold-faced.

Computing times are also given here and can provide first insights into the computational efforts of the different packing heuristics. They are negligibly small for the savings heuristic combined with construction heuristics (on average less than 0.1 seconds). It can be observed, though, that the extension of the placement test leads to an increase in computing times and that the Sav×TA heuristics also require more time. The latter is explicable as the TA heuristics examine *all* potential place-

ment positions for an item whereas the DBLF heuristics terminate as soon as one feasible placement is found. Naturally, Sav×LS_DBLF (on average 0.6 seconds) and Sav×LS_OS (1.67 seconds) also require longer computation times. This is particularly noticeable for small items where Sav×LS_DBLF requires on average almost one second and Sav×LS_OS even more than two seconds. The instances with small items are computationally quite expensive as more items can fit into the loading space and more placement positions need to be tested.

The results support the insights from the experiments on randomly generated routes. The DBLF is identified as the best packing construction heuristic (among the ones presented in Chapters 4.1.2.1 and 4.1.2.2). Therefore, the three packing heuristics DBLF, LS_DBLF and LS_OS are employed in the following experiments.

5.5.1.3 Container loading instances

In order to demonstrate why comparatively simple packing heuristics are chosen for the hybrid solution approach presented in this thesis, the heuristics DBLF, LS_DBLF and LS_OS are applied to well-known CLP instances of Bischoff and Ratcliff (1995) and Davies and Bischoff (1999). For this purpose, the packing heuristics are adapted to the optimization problem (maximization of the volume utilization, see Chapter 4.1.4) and compared to the genetic algorithm presented by Gonçalves and Resende (2012), which proved to be one of the most effective algorithms to date.[35]

The results are listed in Table 5.9. The volume utilizations (∅vol) that are achieved on average and the average computing times (∅ct) per instance (in seconds) are given. The last two columns (GR12) contain the results of Gonçalves and Resende (2012). The best results (with regard to volume utilization) of the packing heuristics presented in this thesis are bold-faced.

The CLP approach of Gonçalves and Resende (2012) clearly outperforms the simpler heuristics achieving an average volume utilization of 92.24 %. In contrast to the previous experiments, LS_DBLF dominates LS_OS in most problem sets. Average volume utilizations of approximately 75.6 %, 79.8 %, and 77.0 % are achieved by the heuristics DBLF, LS_DBLF, and LS_OS, respectively.

The computing times needed by the packing heuristics are very low: 0.01 s (DBLF),

[35] The computer used by Gonçalves and Resende (2012) has the following characteristics: AMD 2.2 GHz Opteron 6-core CPU with Linux (Fedora release 12) operating system.

3.6 s (LS_DBLF), and 8.8 s (LS_OS). In contrast, the GA of Gonçalves and Resende (2012) requires on average 232 s. Hence, despite the higher quality of the GA, it is not suitable to be implemented into a hybrid approach for the 3L-VRPBTW as the computing times are far too high. Within the ALNS, the packing procedure is called several tens of thousands of times during the search and, thus, cannot afford to apply a heuristic that needs more than a few seconds. Even if the computational effort of the GA was drastically decreased (i.e. by reducing the number of iterations), it would still be too high for using the GA in an integrated solution approach for a 3L-VRPBTW.

Table 5.9: Comparison of packing heuristics; average volume utilizations and computing times for CLP instances of Bischoff and Ratcliff (1995) and Davies and Bischoff (1999)

instance set	**no. of item types**	**DBLF**		**LS_DBLF**		**LS_OS**		**GR12**	
		∅vol [%]	∅ct [s]	∅vol [%]	∅ct [s]	∅vol [%]	∅ct [s]	∅vol [%]	∅ct [s]
BRD_01	3	79.02	0.02	81.56	14.07	**84.02**	4.55	94.34	*n.a.*
BRD_02	5	79.02	0.01	82.61	3.41	**83.31**	2.43	94.88	*n.a.*
BRD_03	8	79.62	0.01	**83.41**	2.89	82.19	4.14	95.05	*n.a.*
BRD_04	10	79.32	0.01	**83.23**	2.69	81.26	5.66	94.75	*n.a.*
BRD_05	12	79.29	0.01	**83.30**	2.82	80.72	5.89	94.58	*n.a.*
BRD_06	15	78.07	0.01	**82.66**	2.67	80.15	6.25	94.39	*n.a.*
BRD_07	20	77.32	0.01	**81.92**	2.38	78.90	6.85	93.74	*n.a.*
BRD_08	30	76.01	0.01	**80.77**	2.63	77.49	8.49	92.65	*n.a.*
BRD_09	40	74.60	0.01	**79.39**	2.56	75.86	8.99	91.90	*n.a.*
BRD_10	50	73.88	0.01	**78.60**	2.78	74.38	10.63	91.28	*n.a.*
BRD_11	60	73.25	0.01	**77.73**	2.75	73.14	10.93	90.39	*n.a.*
BRD_12	70	72.42	0.01	**76.75**	2.99	72.22	13.17	89.81	*n.a.*
BRD_13	80	71.16	0.01	**75.94**	3.04	71.50	14.13	89.27	*n.a.*
BRD_14	90	71.07	0.01	**75.18**	2.92	70.70	14.52	88.57	*n.a.*
BRD_15	100	69.67	0.01	**74.42**	3.24	69.28	15.07	87.96	*n.a.*
total		75.58	0.01	**79.83**	3.59	77.01	8.78	92.24	232

∅: average, ct: computing time per instance, *n.a.*: not available, vol: volume utilization.
The maximum volume utilizations per line among DBLF, LS_DBLF and LS_OS are bold-faced.

5.5.2 Hybrid algorithm

In the following, seven problem variants (3L-VRPTW, 3L-VRPCB, 3L-VRPCBTW, 3L-VRPMB, 3L-VRPMBTW, 3L-VRPSDP, 3L-VRPSDPTW), each with two different loading approaches (see Chapter 2.1), are considered. In Chapter 5.5.2, only the constraint set C1 is applied.[36] That is, 14 different extended problem variants are regarded.

[36] Therefore, the fact that C1 is applied is not mentioned any further in Chapter 5.5.2.

Three different variants of the hybrid algorithm, i.e. three alternative packing heuristics (DBLF, LS_DBLF and LS_OS) integrated into the ALNS, are applied to the instances of the mentioned extended problem variants. Hereinafter, the variants of the hybrid algorithm with the different packing heuristics are denoted as ALNS×DBLF, ALNS×LS_DBLF and ALNS×LS_OS.

Each instance of the considered extended problem variants is solved five times by each variant of the hybrid algorithm (ALNS×DBLF, ALNS×LS_DBLF, ALNS×LS_OS). For each algorithm variant, the solution of an instance (of a certain extended problem variant) is in the following defined as the solution (obtained by the respective algorithm variant) that is feasible and that leads to the minimum TTD among the feasible solutions obtained in the five corresponding runs.

As there are no benchmark solutions available for the newly generated instances, the following analyses focus mainly on comparisons of the packing heuristics and loading approaches. In Chapter 5.5.2.1, the results for the different problem variants are presented and examined. The different instance characteristics influence the solutions and the performance of the variants of the hybrid algorithm. The corresponding analyses are provided in Chapter 5.5.2.2. In particular, the item size can have a huge impact on the performance of the hybrid algorithms. Therefore, the results are sometimes presented for instances with small and large items separately. In addition, conclusions will be drawn about different backhaul models (Chapter 5.5.2.3), the presence of time windows (Chapter 5.5.2.4) and about the application of solution approaches for VRPs with loading constraints (Chapter 5.5.2.5).

In some cases, the hybrid algorithm is not able to generate a feasible solution. For each analysis, only those instances are considered that are *solved sufficiently*, i.e. a feasible solution could be obtained at least once (in the five conducted runs) by all problem and procedure settings that are compared. For example, if an analysis aims at comparing the variants of the hybrid algorithm for a certain extended problem variant, only those instances are taken into account for which a feasible solution is provided at least once (in five runs) *by each hybrid algorithm*. If the performance of loading approaches is evaluated, instances are considered that are solved at least once by each hybrid algorithm variant and for each of the compared loading approaches for the respective problem variant. The corresponding numbers are provided in the appendix (Table C.1).

Comprehensive results for all problem variants are given in Appendix C.

5.5.2.1 Results for 3L-VRPBTW variants

3L-VRP with time windows

In Table 5.10, the results are presented for the variants of the hybrid algorithm with different packing heuristics for the 3L-VRPTW and the loading approaches rear and side loading. For each packing heuristic and loading approach (and summarized for both approaches), the average TTDs (∅TTD) of the solutions of the instances are given. In order to evaluate the quality of the algorithms, the solutions provided by ALNS×DBLF serve as benchmarks. The average deviations (∅dev) of the TTDs obtained by the variants of the hybrid algorithm from the benchmarks are included in Table 5.10. Moreover, the average computing times per instance and run in seconds (∅ct) and the average numbers of iterations (per instance and run) conducted until the algorithm is terminated (∅iter) are listed. In the following, the tables have the same general structure. Depending on the relevance for the analyses, further key performance indicators (KPIs) may be used, e.g. the average number of iterations per second.

Table 5.10: Comparison of hybrid ALNS algorithms with different packing heuristics; extended problem variants (3L-VRPTW, RL/SL, C1); average TTDs, deviations from benchmark, computing times, numbers of iterations; separated by loading approach

		ALNS		
pack	**KPI**	**rear loading**	**side loading**	**total**
DBLF	∅TTD*[DU]	1,230.01	1,160.65	1,196.02
	∅dev[%]	0.00	0.00	0.00
	∅ct[s]	797.57	719.90	759.51
	∅iter	14,129.77	14,537.16	14,329.42
LS_DBLF	∅TTD[DU]	1,136.52	1,086.23	1,111.88
	∅dev[%]	**−4.63**	**−4.18**	**−4.41**
	∅ct[s]	1,511.79	1,459.02	1,485.93
	∅iter	12,096.06	12,008.50	12,053.15
LS_OS	∅TTD[DU]	1,141.92	1,094.13	1,118.50
	∅dev[%]	−4.18	−3.63	−3.91
	∅ct[s]	1,816.31	1,909.63	1,862.05
	∅iter	10,745.00	9,914.98	10,338.23

∅: average, *: benchmark, ct: computing time per instance and run, dev: deviation from TTDs of benchmarks, DU: distance units, iter: total number of iterations, KPI: key performance indicator, pack: integrated packing heuristic, TTD: total travel distance.
Minimum average deviations per column are bold-faced.

In general, the hybrid algorithms with the LS-based packing heuristics outperform the hybrid algorithm with the simple DBLF heuristic. The average devi-

ations, i.e. improvements, from the benchmarks amount to −4.41 % (ALNS×LS_DBLF) and −3.91 % (ALNS×LS_OS) over both loading approaches. Furthermore, ALNS×DBLF requires considerably shorter computing times on average (760 seconds) than ALNS×LS_DBLF (1,486 seconds) and ALNS×LS_OS (1,862 seconds). ALNS×DBLF conducts (on average) more iterations (14,329) than ALNS×LS_DBLF (12,053) and ALNS×LS_OS (10,338).

Thus, the results indicate a better average quality of ALNS×LS_DBLF compared to ALNS×DBLF and ALNS×LS_OS. In addition, it be concluded that ALNS×LS_DBLF requires less computational effort than ALNS×LS_OS but noticeably more than ALNS×DBLF. The latter results from the low complexity of the packing construction heuristic DBLF compared to the LS-based heuristics. The performance (concerning solution quality and computing time) of the individual hybrid algorithms depends heavily on the instance characteristics, like item size or number of customers. These findings are analysed below (Chapter 5.5.2.2).

The packing heuristics work equally well for both considered loading approaches and the computing times for both approaches differ only marginally. It will, thus, be more interesting to compare the performance of the loading approaches themselves instead of the performance of the packing heuristics with the individual loading approaches.

Aggregated results for the 3L-VRPTW are presented in Table 5.11. The base for this analysis is the following: Pairs of instances are compared that are derived from the same core instances and differ only in the loading approach. The comparisons are conducted for each variant of the hybrid algorithm separately. That is, the solutions obtained by a certain algorithm variant for each instance of a pair are compared. The solutions of the rear loading instances serve as benchmarks. Per instance and variant of the hybrid algorithm, the deviation of the respective TTD from the corresponding benchmark TTD is determined. As the results do not differ noticeably among the individual packing heuristics, the deviations are averaged over the variants of the hybrid algorithm in Table 5.11.

The average TTDs (⌀TTD), average deviations from the benchmark TTDs (⌀dev) and the average numbers of used vehicles ($\varnothing v_{used}$) are given. As mentioned above, the item size impacts the performance of the hybrid algorithms. Therefore, the results are presented in Table 5.11 for instances with small and large items separately

and together.[37] The minimum deviations per item size are bold-faced.

Table 5.11: Comparison of loading approaches; extended problem variants (3L-VRPTW, RL/SL, C1); average TTDs, deviations from benchmark, numbers of used vehicles; separated by item size, aggregated over the variants of the hybrid algorithm

		ALNS		
load	**KPI**	**small items**	**large items**	**total**
rear loading	∅TTD*[DU]	791.64	1,741.28	1,108.19
	∅dev[%]	**0.00**	0.00	0.00
	$\varnothing v_{used}$	6.78	22.58	12.05
side loading	∅TTD[DU]	792.32	1,733.69	1,106.11
	∅dev[%]	0.16	**−0.37**	**−0.02**
	$\varnothing v_{used}$	6.80	22.35	11.98

∅: average, *: benchmark, dev: deviation from TTDs of benchmarks, DU: distance units, KPI: key performance indicator, load: applied loading approach, TTD: total travel distance, v_{used}: number of used vehicles.
Minimum average deviations per column are bold-faced.

In total, the differences between the loading approaches are marginal. The TTDs provided by side loading deviate on average only −0.02 % from the TTDs obtained by rear loading and the average numbers of used vehicles differ by only 0.07. It can be observed, though, that the side loading approach is a bit more beneficial for problems with large items (average deviation from rear loading TTDs: −0.37 %). An explanation may be that large items can be arranged more easily with a longer loading side. The rear loading approach leads to slightly better results for the instances with small items. However, with an average deviation of 0.16 %, the differences are minor.

3L-VRP with clustered backhauls and with and without time windows

The results for the 3L-VRPCB(TW) and the loading approaches rear and side loading are summarized in Table 5.12. The method of analysis is analogue to the one presented for Table 5.10. The same performance indicators as above are used and the solutions provided by ALNS×DBLF serve as benchmarks. The table contains the results for both problem variants with and without time windows together.

The overall results resemble those of the 3L-VRPTW. For both loading approaches, the best results are on average obtained by ALNS×LS_DBLF. The average deviations from the TTDs provided by ALNS×DBLF amount to −4.31 % for the rear loading approach and to −3.69 % for the side loading approach, respectively. The

[37] This is also done frequently in the remainder of Chapter 5.5.2. Further observations about the impact of the item size are provided in Chapter 5.5.2.2.

average results of ALNS×LS_OS are only slightly worse (rear loading: −3.89 %, side loading: −3.27 %). Also with respect to the computational efforts, the results are similar to those of the 3L-VRPTW. On average, ALNS×DBLF requires less computing time (845 seconds) and conducts more iterations (13,391) than ALNS×LS_DBLF (1,666 seconds; 10,718 iterations) and ALNS×LS_OS (1,989 seconds; 9,038 iterations).

Table 5.12: Comparison of hybrid ALNS algorithms with different packing heuristics, extended problem variants (3L-VRPCB(TW), RL/SL, C1); average TTDs, deviations from benchmark, computing times, numbers of iterations; separated by loading approach

		ALNS		
pack	**KPI**	**rear loading**	**side loading**	**total**
DBLF	∅TTD[DU]	1,104.08	1,047.00	1,076.25
	∅dev[%]	0.00	0.00	0.00
	∅ct[s]	885.51	803.36	845.46
	∅iter	13,228.26	13,561.62	13,390.81
LS_DBLF	∅TTD[DU]	1,029.28	988.71	1,009.50
	∅dev[%]	**−4.31**	**−3.69**	**−4.01**
	∅ct[s]	1,679.95	1,651.87	1,666.26
	∅iter	10,906.07	10,522.02	10,718.81
LS_OS	∅TTD[DU]	1,032.61	992.72	1,013.16
	∅dev[%]	−3.89	−3.27	−3.59
	∅ct[s]	1,948.01	2,031.17	1,988.56
	∅iter	9,498.33	8,554.04	9,037.89

∅: average, *: benchmark, ct: computing time per instance and run, dev: deviation from TTDs of benchmarks, DU: distance units, iter: total number of iterations, KPI: key performance indicator, pack: integrated packing heuristic, TTD: total travel distance.
Minimum average deviations per column are bold-faced.

As before, no loading approach (rear or side loading) is clearly dominant. Therefore, a listing and analysis of results is ommited here. Detailed results are provided in the appendix (Table C.9).

3L-VRP with mixed backhauls and with and without time windows

Table 5.13 contains the results for the 3L-VRPMB(TW) and the loading approaches LSP and side loading. The previously introduced KPIs are used and the results for the problem variants with and without time windows are presented jointly again. The solutions obtained by ALNS×DBLF and ALNS×DBLF^{SL}, respectively, serve as benchmarks for the individual loading approaches.

In order to realize the simultaneous transport of linehaul and backhaul items, the loading approaches loading space partition (LSP) and side loading are ap-

plied. In contrast to the side loading approach applied to the 3L-VRPTW and 3L-VRPCB(TW), which is implemented by swapping the loading space dimensions length and width, the side loading approach described in Chapter 4.1.5 is utilized for the simultaneous transport. The linehaul and backhaul items are loaded at opposing sides of the loading space (e.g. linehaul items at the driver's cabin and backhaul items at the rear). Additional checks are necessary in order to avoid the overlapping of linehaul and backhaul items at any stage of the route. The DBLF heuristic is modified for this loading approach with respect to the sorting of the possible placement positions ($DBLF^{SL}$, see Chapter 4.1.2.1).

Table 5.13: Comparison of hybrid ALNS algorithms with different packing heuristics, extended problem variants (3L-VRPMB(TW), LSP/SL, C1); average TTDs, deviations from benchmark, computing times, numbers of iterations; separated by loading approach

		ALNS		
pack	**KPI**	**LSP**	**side loading**	**total**
DBLF**	∅TTD*[DU]	1,002.33	911.94	951.93
	∅dev[%]	0.00	0.00	0.00
	∅ct[s]	686.36	1,114.86	925.30
	∅iter	18,542.98	13,316.69	15,628.69
LS_DBLF**	∅TTD[DU]	931.92	866.32	895.34
	∅dev[%]	**−4.49**	−3.56	−3.97
	∅ct[s]	1,330.52	1,818.16	1,602.44
	∅iter	12,379.84	12,022.68	12,180.68
LS_OS	∅TTD[DU]	928.28	862.65	891.68
	∅dev[%]	−4.35	**−3.97**	**−4.14**
	∅ct[s]	1,971.02	2,080.03	2,031.81
	∅iter	8,875.09	10,236.94	9,634.48

∅: average, *: benchmark, **: $DBLF^{SL}$ and LS_DBLF^{SL}, respectively, are employed for the side loading approach, ct: computing time per instance and run, dev: deviation from TTDs of benchmarks, DU: distance units, iter: total number of iterations, KPI: key performance indicator, LSP: loading space partition, pack: integrated packing heuristic, TTD: total travel distance.
Minimum average deviations per column are bold-faced.

For the LSP approach, the best results are obtained by ALNS×LS_DBLF (average deviation from benchmark: −4.49 %), which slightly outperforms ALNS×LS_OS (−4.35 %). The results for the side loading approach differ a bit from the results of the LSP. The best results are obtained by ALNS×LS_OS (average deviation from benchmark: −3.97 %). With an average deviation from the TTDs of the benchmark solutions of −3.56 %, the solutions obtained by ALNS×LS_DBLF^{SL} are only slightly worse.

Naturally, ALNS×DBLF (and ALNS×$DBLF^{SL}$) requires the shortest computing

times. Moreover, noticeably longer computing times are needed for the side loading approach than for the LSP approach. For example, one run of ALNS×DBLF^{SL} with the side loading approach takes about seven minutes longer than one run of ALNS×DBLF with LSP. The generation of two packing patterns (for linehaul and backhaul items separately, see Chapter 4.1.5) is required by both approaches. In addition, it must be considered that more items can be packed using the side loading approach as the whole loading space can be utilized for both linehaul and backhaul items. In extreme cases, two packing patterns with high volume utilization need to be produced, whereas the items included in the individual patterns of the LSP cannot exceed half the loading space volume. More items to be packed increase the computational efforts for solving the packing subproblems.

As the overall results of ALNS×LS_DBLF are often better than those of ALNS×-LS_OS for previously examined loading variants, it can be assumed that the LS_OS heuristic is more suitable (compared to LS_DBLF^{SL}) for the side loading approach as it is implemented here. No modification is incorporated in LS_OS for coping with the applied side loading approach (apart from the consideration of the modified LIFO constraint). Nevertheless, the results of LS_OS show similar deviations from the benchmark TTDs as in previous experiments. One reason could be that the change in the item sequence due to the LS-framework is capable of coping with the specifications of the side loading approach. In order to test whether this is also true for the other LS-based packing heuristic, ALNS×LS_DBLF is also used for the side loading. That is, the original DBL sorting rule is applied to the DBLF integrated into the local search. However, the results of ALNS×LS_DBLF are clearly worse than the results of ALNS×LS_DBLF^{SL} and ALNS×LS_OS. The average deviations from the TTDs of the benchmark solutions amount to only -1.6 %, which is about two percentage points less than the deviations of the other LS variants.[38]

In conclusion, the modification to the DBLF approach is not only beneficial to the construction heuristic DBLF, which cannot avoid the building of gaps in the loading plan (see Chapter 4.1.5), but also to the LS-based variant (LS_DBLF). As the best results are obtained by LS_OS, a construction heuristic working according to a deepest-left-bottom procedure (like the OS heuristic; see Chapter 4.1.2.3) appears to be most suitable for this kind of loading approach.

Subsequently, the quality of the two loading approaches is evaluated. The method of

[38] These results are not included in Table 5.13.

analysis is analogue to the method presented for the 3L-VRPTW. The solutions of the LSP instances serve as benchmarks (for each variant of the hybrid algorithm separately). In Table 5.14, the corresponding results are summarized for all algorithm variants.

The average TTDs (∅TTD) are given and the average deviations (∅dev) between the TTDs of the individual loading approaches from the benchmark TTDs are presented. In addition, the average numbers of used vehicles ($\varnothing v_{used}$) for both loading approaches are listed. The results are separated by the item sizes. The solutions obtained by ALNS×DBLF^{SL} (ALNS×LS_DBLF^{SL}; ALNS×LS_OS) for side loading instances are compared to the solutions of ALNS×DBLF (ALNS×LS_DBLF; ALNS×LS_OS) for the corresponding LSP instances.

Table 5.14: Comparison of loading approaches; extended problem variants (3L-VRPMB(TW), LSP/SL, C1); average TTDs, deviations from benchmark, numbers of used vehicles; separated by item size and in total, aggregated over the variants of the hybrid algorithm

		ALNS		
load	**KPI**	**small items**	**large items**	**total**
LSP	∅TTD*[DU]	664.35	1,785.97	954.18
	∅dev[%]	0.00	0.00	0.00
	$\varnothing v_{used}$	5.00	22.86	9.61
SL	∅TTD[DU]	641.43	1,251.76	799.14
	∅dev[%]	**−3.78**	**−29.02**	**−10.31**
	$\varnothing v_{used}$	4.32	13.23	6.62

∅: average, *: benchmark, dev: deviation from TTDs of benchmarks, DU: distance units, KPI: key performance indicator, load: applied loading approach, LSP: loading space partition, SL: side loading, TTD: total travel distance, v_{used}: number of used vehicles.
Minimum average deviations per column are bold-faced.

The side loading provides better results in almost all instance classes. The item size has the most noticeable impact on the differences between the two loading approaches. Whereas the average deviation of the TTDs provided by side loading from the TTDs of the LSP solutions amounts to −3.78 % for instances with small items, an average deviation of −29.02 % is obtained for instances with large items. That is, the side loading approach is considerably more beneficial (compared to LSP) if the items are large. Small items are easier to pack into the separated loading space and they also make up less volume than the large items as the total number of items remains constant. As the usable loading space is restricted in the case of the LSP approach, each vehicle leaves the depot with an (at least) half empty loading space

and returns to the depot with an (at least) half empty loading space. In contrast, the whole loading space can be used when applying the side loading approach. Accordingly, longer routes can be generated and fewer vehicles are needed with the side loading approach. On average, about three vehicles can be saved.

3L-VRP with simultaneous delivery and pickup and with and without time windows
The results of the 3L-VRPSDP(TW) resemble those of the experiments with the 3L-VRPMB(TW). They are summarized in Table 5.15 and include both problem variants with and without time windows. The loading approaches LSP and side loading are regarded and the solutions provided by ALNS×DBLF and ALNS×DBLF^{SL}, respectively, serve as benchmarks. The key performance indicators introduced above are used.

Table 5.15: Comparison of hybrid ALNS algorithms with different packing heuristics; extended problem variants (3L-VRPSDP(TW), LSP/SL, C1); average TTDs, deviations from benchmark, computing times, numbers of iterations; separated by loading approach

		ALNS		
pack	**KPI**	**LSP**	**side loading**	**total**
DBLF**	∅TTD*[DU]	1,098.10	971.14	1,033.49
	∅dev[%]	0.00	0.00	0.00
	∅ct[s]	640.93	1,120.77	885.14
	∅iter	15,295.37	14,173.39	14,724.35
LS_DBLF**	∅TTD[DU]	1,051.41	945.98	997.75
	∅dev[%]	−3.01	−1.71	−2.35
	∅ct[s]	1,456.40	1,651.67	1,555.78
	∅iter	12,096.01	13,408.19	12,763.82
LS_OS	∅TTD[DU]	1,047.97	928.90	987.37
	∅dev[%]	**−3.07**	**−3.73**	**−3.40**
	∅ct[s]	2,417.48	2,151.35	2,282.04
	∅iter	7,160.96	10,133.70	8,673.90

∅: average, *: benchmark, **: DBLF^{SL} and LS_DBLF^{SL}, respectively, are employed for the side loading approach, ct: computing time per instance and run, dev: deviation from TTDs of benchmarks, DU: distance units, iter: total number of iterations, KPI: key performance indicator, LSP: loading space partition, pack: integrated packing heuristic, TTD: total travel distance.
Minimum average deviations per column are bold-faced.

Regarding the LSP, both variants of the hybrid algorithm with LS-based packing heuristics outperform ALNS×DBLF. Here, ALNS×LS_OS provides slightly better results than ALNS×LS_DBLF, but the differences are marginal with average deviations from the benchmarks of −3.01 % (ALNS×LS_DBLF) and −3.07 % (ALNS×-LS_OS). Applying the side loading approach, the results differ more from the results

of the 3L-VRPMB(TW). As before, the best results are achieved by ALNS×LS_OS (average deviation from benchmarks of −3.73 %). However, in the previous experiments the differences between ALNS×LS_DBLF^{SL} and ALNS×LS_OS are rather small. In contrast, the deviations from the benchmarks differ by more than two percentage points here and ALNS×LS_OS is the dominant heuristic in almost all instance classes.

ALNS×LS_DBLF with the original DBL sorting of the positions is also applied to the 3L-VRPSDP(TW) with side loading. As in the experiments of the 3L-VRPMB(TW), it leads to worse results than the other LS-based heuristics. Surprisingly, though, it even results in (on average) worse solutions than ALNS×DBLF^{SL} with an average deviation from the benchmark TTDs of 1.25 %.[39] As before, it can be concluded that either the adapted approach DBLF^{SL} or the open space heuristic work best for the simultaneous transport of linehaul and backhaul items with side loading.

Comparing the different loading approaches, the conclusions are not completely identical to those drawn in the case of the problem variant with mixed backhauls. The results for the 3L-VRPSDP(TW) are presented in Table 5.16, which is analogous to Table 5.14. The table contains the average TTDs (⌀TTD) of the instance solutions for both loading approaches and the average deviations (⌀dev) of these TTDs from the TTDs of the benchmarks (solutions obtained for the LSP instances). Moreover, the average numbers of used vehicles ($\varnothing v_{used}$) are presented. These results are aggregated over all variants of the hybrid algorithm.

The average deviation of the TTDs of the side loading from the TTDs of the LSP solutions over all packing heuristics amounts to −0.53 % for instances with small items and to −18.96 % for instances with large items. Whereas the average numbers of used vehicles are similar for instances with small items (LSP: 4.76, side loading: 4.64), they differ substantially for instances with large items (LSP: 23.27, side loading: 16.24).

In the case of the 3L-VRPMB(TW), the side loading is almost exclusively dominant over all instance classes. Here, a slightly different picture is shown. For instances with small items, no approach is clearly dominant. Small items can be packed more easily into the separated loading space. Therefore, the LSP is more applicable for small than for large items. Furthermore, the nature of the problem variant leads

[39] These results are not included in Table 5.15.

to different planning restrictions. In the case of the 3L-VRPMB(TW), routes could be formed containing, for example, a large number of linehaul items and only very few backhaul items, or even no items of one type. In contrast, items of both types *have* to be transported on every route if a 3L-VRPSDP(TW) instance is solved. The proportions are prescribed by the instance parameters. Thus, in particular in the case of small items, the LSP approach can be well-suited for the problem and it can also benefit from the lower requirements in computation times. For large items, the side loading approach is clearly more suitable as noticeably shorter TTDs are obtained and less vehicles are required.

Table 5.16: Comparison of loading approaches; extended problem variants (3L-VRPSDP(TW), LSP/SL, C1); average TTDs, deviations from benchmark, numbers of used vehicles; separated by item size and in total, aggregated over the variants of the hybrid algorithm

		ALNS		
load	**KPI**	**small items**	**large items**	**total**
LSP	∅TTD*[DU]	652.32	1,752.32	1,063.62
	∅dev[%]	0.00	0.00	0.00
	$\varnothing v_{used}$	4.76	23.27	11.68
SL	∅TTD[DU]	649.89	1,394.14	928.18
	∅dev[%]	**−0.53**	**−18.96**	**−7.42**
	$\varnothing v_{used}$	4.64	16.24	8.98

∅: average, *: benchmark, dev: deviation from TTDs of benchmarks, DU: distance units, KPI: key performance indicator, load: applied loading approach, LSP: loading space partition, SL: side loading, TTD: total travel distance, v_{used}: number of used vehicles.
Minimum average deviations per column are bold-faced.

Comparisons and analyses across problem variants

Table 5.17 contains the summarized results comparing the variants of hybrid algorithm for all considered 3L-VRPBTW variants and related loading approaches. As before, the solutions provided by ALNS×DBLF or ALNS×DBLF^{SL} (depending on the loading approach) serve as benchmark solutions.[40] The average TTDs (∅TTD) provided by the variants of the hybrid algorithm are given for each problem variant and the corresponding loading approaches. In addition, the average deviations (∅dev) from the TTDs of the benchmark solutions, the average computing times

[40] That is, ALNS×DBLF is considered for the rear loading and LSP instances and the side loading instances of the problem variants 3L-VRPTW and 3L-VRPCB(TW). ALNS×DBLF^{SL} is employed for the side loading instances of the problem variants 3L-VRPMB(TW) and 3L-VRPSDP(TW).

per run and instance (∅ct), the average number of conducted iterations (∅iter) and the shares of runs terminated by the time limit (abort) are given.

With the LS-based packing heuristics, the hybrid algorithm is able to find considerably better solutions than with the DBLF heuristic. The differences between the variants of the hybrid algorithm with the LS-based heuristics are marginal. The average deviations from the benchmark TTDs amount to −3.73 % for ALNS×LS_DBLF and to −3.72 % for ALNS×LS_OS over all problem variants and loading approaches. The results suggest that ALNS×LS_DBLF provides better results for the loading approaches that are implemented as rear loading approaches. These include the "standard" rear loading approach, the LSP (where the vehicles are also rear loaded), but also the side loading approach for the 3L-VRPTW and the 3L-VRPCB(TW). The latter is implemented like the rear loading approach by swapping length and width of the loading space. On average, ALNS×LS_DBLF obtains the best solutions for the instances of these extended problem variants. The only exception is the problem variant of 3L-VRPSDP(TW) and the LSP approach. However, the average differences between ALNS×LS_DBLF and ALNS×LS_OS are marginal for this variant (average benchmark deviations of ALNS×LS_DBLF: −3.01 %, ALNS×LS_OS: −3.07 %). ALNS×LS_OS obtains the best solutions for the side loading approach for the 3L-VRPMB(TW) and 3L-VRPSDP(TW), where the modified LIFO constraint is taken into account (cf. Chapter 4.1.5). In case of the 3L-VRPSDP(TW), noticeable differences between ALNS×LS_OS and ALNS×LS_DBLF^{SL} occur. Whereas the TTDs provided by ALNS×LS_OS deviate on average −3.73 % from the benchmark TTDs of ALNS×DBLF^{SL}, the average deviations of the TTDs provided by ALNS×LS_DBLF^{SL} from the benchmark TTDs amount to (only) −1.71 %.

ALNS×DBLF is the fastest approach among the presented ones providing solutions, on average, within approximately 852 seconds. ALNS×LS_DBLF (ca. 1,595 seconds) and ALNS×LS_OS (ca. 2,034 seconds) require about twice as much computing time. The larger computational requirements of the LS-based approaches are reflected in the average numbers of conducted iterations and the shares of runs that are terminated by the time limit. Less runs are aborted by ALNS×DBLF (ca. 14.7 %) than by ALNS×LS_DBLF (ca. 40.2 %) or ALNS×LS_OS (ca. 55.7 %). Until the termination, approximately 14,290 (ALNS×DBLF), 11,697 (ALNS×LS_DBLF) and 9,335 iterations (ALNS×LS_OS) are conducted on average.

Table 5.17: Comparison of packing heuristics for different 3L-VRPBTW variants and loading approaches; average TTDs, deviations from benchmark, computing times, numbers of iterations, share of runs terminated by time limit; separated by problem variant and loading approach

		ALNS								
		3L-VRPTW		**3L-VRPCB(TW)**		**3L-VRPMB(TW)**		**3L-VRPSDP(TW)**		
pack	**KPI**	RL	SL	RL	SL	LSP	SL	LSP	SL	**total**
DBLF**	∅TTD*[DU]	1,230.01	1,160.65	1,104.08	1,047.00	1,002.33	911.94	1,098.10	971.14	1,067.05
	∅dev[%]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	∅ct[s]	797.57	719.90	885.51	803.36	686.36	1,114.86	640.93	1,120.77	852.14
	∅iter	14,129.77	14,537.16	13,228.26	13,561.62	18,542.98	13,316.69	15,295.37	14,173.39	14,289.85
	abort[%]	15.68	12.82	16.32	13.36	6.89	21.20	6.55	22.14	14.68
LS_DBLF**	∅TTD[DU]	1,136.52	1,086.23	1,029.28	988.71	931.92	866.32	1,051.41	945.98	1,005.34
	∅dev[%]	**−4.63**	**−4.18**	**−4.31**	**−3.69**	**−4.49**	−3.56	−3.01	−1.71	**−3.73**
	∅ct[s]	1,511.79	1,459.02	1,679.95	1,651.87	1,330.52	1,818.16	1,456.40	1,651.67	1,594.85
	∅iter	12,096.06	12,008.50	10,906.07	10,522.02	12,379.84	12,022.68	12,096.01	13,408.19	11,696.63
	abort[%]	38.55	38.34	43.22	43.33	37.06	42.73	39.06	33.08	40.20
LS_OS	∅TTD[DU]	1,141.92	1,094.13	1,032.61	992.72	928.28	862.65	1,047.97	928.90	1,005.22
	∅dev[%]	−4.18	−3.63	−3.89	−3.27	−4.35	**−3.97**	**−3.07**	**−3.73**	−3.72
	∅ct[s]	1,816.31	1,909.63	1,948.01	2,031.17	1,971.02	2,080.03	2,417.48	2,151.35	2,033.59
	∅iter	10,745.00	9,914.98	9,498.33	8,554.04	8,875.09	10,236.94	7,160.96	10,133.70	9,334.59
	abort[%]	46.90	50.68	53.28	56.77	59.62	53.17	72.24	54.62	55.69

∅: average, *: benchmark, **: (LS_)DBLF [for (3L-VRPTW,RL/SL,C1), (3L-VRPCB(TW),RL/SL,C1), (3L-VRPMB(TW)/3L-VRPSDP(TW),LSP,C1)] or (LS_)DBLF^{SL} [for (3L-VRPMB(TW)/3L-VRPSDP(TW), SL, C1)], abort: share of runs aborted by time limit, ct: computing time per instance and run, dev: deviation from TTDs of benchmarks, DU: distance units, iter: total number of iterations, KPI: key performance indicator, LSP: loading space partition, RL: rear loading, SL: side loading, TTD: total travel distance.
Minimum average deviations per column are bold-faced.

Furthermore, the problem variants can be compared with respect to the computing times required for solving them with the hybrid algorithm. The average computing times per instance and run (in seconds) for the seven problem variants are presented in summarized form in Table 5.18, subdivided by the packing heuristics employed in the hybrid algorithm. Only instances with 200 items are considered as those with 400 items are not regarded for the 3L-VRPMB(TW) and 3L-VRPSDP(TW). Including them into the analysis would bias the results.

Table 5.18: Comparison of hybrid algorithms with different packing heuristics and for different problem variants; average computing times; separated by applied variant of the hybrid algorithm, limited to $m = 200$ and aggregated over the corresponding loading approaches

	ALNS ($m = 200$)			
problem	**DBLF** ∅ct[s]	**LS_DBLF** ∅ct[s]	**LS_OS** ∅ct[s]	**total** ∅ct[s]
without time windows				
3L-VRPCB	1,062.6	2,290.6	2,627.0	1,993.4
3L-VRPMB	1,406.3	2,319.2	2,700.4	2,142.0
3L-VRPSDP	1,340.0	2,245.1	2,878.0	2,154.4
with time windows				
3L-VRPTW	577.1	1,276.0	1,704.8	1,186.0
3L-VRPCBTW	176.1	557.4	1,126.6	620.0
3L-VRPMBTW	433.4	869.5	1,348.2	883.7
3L-VRPSDPTW	428.0	862.9	1,683.1	991.3
total	771.9	1,488.4	2,010.8	1,423.7

∅ct: average computing times per instance and run

The consideration of time windows reduces the computing times. The time windows restrict the possible solutions leading to fewer customers (and, thus, fewer items) per route (see below for further details).

Regarding the different problems, it can be observed that the instances with clustered backhauls (3L-VRPCB(TW)) are solved fastest on average. They contain the most restrictive routing problem. Thus, there are less potential insertion positions to be tested for feasibility in the course of the insertion heuristic. In contrast, the 3L-VRPTW and the 3L-VRPSDP(TW) require the longest computing times. Although the 3L-VRPTW is the simplest of the studied problems with respect to its routing subproblem, it allows for packing plans with many items. As the same basic instances are used for all problem variants (cf. Chapter 5.2.2), it can be concluded that the routes contain similar numbers of customers. In case of the 3L-VRPTW, all items of these customers must be aggregated to one packing pattern, whereas

(up to) two packing patterns are required for the remaining problems. However, less items need to be packed per pattern as they are split into linehaul and backhaul items. Generating two packing patterns for a lower number of items can require less computation time than generating one pattern for a larger number of items. This reduction of the difficulty of the packing problem reflects in the reduced computing times when backhauls are considered.

5.5.2.2 Influence of selected instance characteristics

The performance of the individual approaches depends heavily on the instance characteristics and different versions of the hybrid algorithm (with respect to the applied packing heuristics) are dominant in different problem classes. The influence of the following characteristics is analysed in detail: item size, total number of items, number of customers, time window width, number of item types. As their impacts are similar among all problem variants, the results are mainly analysed based on the results of the 3L-VRPTW. The conclusions can be transferred to the other problem variants as well. In addition, the impact of linehaul share on the benefits of different loading approaches is examined.

Item size

In Table 5.19, the results of the experiments of the hybrid algorithm for the 3L-VRPTW with rear and side loading are separated by the items sizes and aggregated for both loading approaches (as the results do not differ considerably). No instances are available for $n = 20$ and large items (see Chapter 5.2.2). Therefore, only instances with 60 and 100 customers are included in Table 5.19 in order to exclude the influence of the varying numbers of customers.[41] The solutions provided by ALNS×-DBLF serve as benchmarks. The average TTDs (⌀TTD) of the instance solutions, the average deviations (⌀dev) from the benchmark TTDs, average computing times (⌀ct) and numbers of conducted iterations (⌀iter) are listed. In addition, the shares of runs terminated due to the time limit (abort) are given.

If the items are small, the solution quality of the the hybrid algorithms does not vary much. Interestingly, ALNS×DBLF outperforms ALNS×LS_DBLF and ALNS×-

[41] For example, the computing times could not be compared. 60 or 100 customers are considered in the instances with large items, but 20, 60 and 100 customers are included in the instances with small items. The instances with 20 customers, which require less computing times, would bias the results in favour of the instances with small items.

LS_OS, which provide solutions whose TTDs deviate on average 0.21 % and 0.79 %, respectively, from the TTDs obtained by ALNS×DBLF. If the items are large, the solutions of ALNS×LS_DBLF and ALNS×LS_OS are clearly better than those of ALNS×DBLF resulting in average TTD deviations of −11.14 % and −10.67 %, respectively. The computing times required by ALNS×LS_DBLF and ALNS×LS_OS for solving instances with small and large items do not differ considerably. Solving an instance with small items takes on average about 28.8 (34.2) minutes if ALNS×LS_DBLF (ALNS×LS_OS) is employed. For instances with large items, approximately 29.5 (40.6) minutes are needed by ALNS×LS_DBLF (ALNS×LS_OS). A different picture is drawn by ALNS×DBLF. Solving an instance with small items takes about ten times longer (26.1 minutes) than solving an instance with large items (2.5 minutes).

Table 5.19: Comparison of hybrid ALNS algorithms with different packing heuristics; extended problem variants (3L-VRPTW, RL/SL, C1); average TTDs, deviations from benchmark, computing times, numbers of iterations, proportions of prematurely aborted runs; separated by item size, restricted to $n \in \{60, 100\}$ and aggregated over both loading approaches

		ALNS ($n \in \{60, 100\}$)	
pack	**KPI**	**small items**	**large items**
DBLF	∅TTD[DU]	959.77	1,944.36
	∅dev[%]	**0.00**	0.00
	∅ct[s]	1,563.53	151.39
	∅iter	11,843.77	21,039.56
	abort[%]	31.71	0.00
LS_DBLF	∅TTD[DU]	961.78	1,709.89
	∅dev[%]	0.21	**−11.14**
	∅ct[s]	1,727.45	1,767.95
	∅iter	9,711.13	19,894.71
	abort[%]	39.29	26.75
LS_OS	∅TTD[DU]	966.79	1,721.44
	∅dev[%]	0.79	−10.67
	∅ct[s]	2,054.14	2,435.83
	∅iter	8,631.10	16,595.79
	abort[%]	45.67	46.91

∅: average, *: benchmark, abort: share of runs aborted by time limit, ct: computing time per instance and run, dev: deviation from TTDs of benchmarks, DU: distance units, iter: total number of iterations, KPI: key performance indicator, pack: integrated packing heuristic, TTD: total travel distance.
Minimum average deviations per column are bold-faced.

Thus, the results suggest that the item sizes have a huge impact on the performance of the hybrid algorithm. Yet, different tendencies can be observed, like the differences in the relative solution quality of the algorithm or the fact that ALNS×DBLF

requires noticeably less computing time for instances with large items than for those with small items. As opposed to this, ALNS×LS_DBLF and ALNS×LS_DBLF need, on average, less time for solving instances with small items.

Three factors can be identified that are influenced by the item size. First, generating *one* packing pattern using a construction heuristic can be done faster if the items are large. The reason for this is that normally less items can fit into the loading space and need to be packed. This can be an explanation for the considerably shorter computing times for large item instances if ALNS×DBLF is applied. Moreover, it reflects in the large numbers of iterations that are conducted by each variant of the hybrid algorithm if the items are large. About twice as many ALNS iterations are performed on instances with large items than on instances with small items, although the computing times and shares of aborted runs for large items are either considerably shorter/lower (ALNS×DBLF) or do not differ much (ALNS×LS_DBLF, ALNS×LS_OS). This can be one explanation for the different deviations from the benchmark TTDs depending on the item size. The smaller numbers of conducted iterations in the case of small items can result in solutions of lower quality as the initial solutions could not be improved sufficiently. ALNS×-DBLF achieves to conduct more iterations and might, therefore, have an advantage over ALNS×LS_DBLF and ALNS×LS_OS. In the case of large items, ALNS×DBLF also performs more iterations. Yet, the numbers of iterations conducted by ALNS×-LS_DBLF and ALNS×LS_OS is presumably large enough to obtain solutions of high quality. In this case, other factors influence the differences in the solution quality (see below). Furthermore, the lower computational requirements for packing large items cause that one iteration of the local search packing heuristics can be performed faster.

The second factor is, though, that small items tend to make the packing problem easier. That is, it is more probable to find a feasible position for each item. This can be one reason why the differences in the solution quality between ALNS×DBLF and ALNS×LS_DBLF/ALNS×LS_OS is rather small. As the packing problem is easier, the LS-based approaches do not provide valuable advantages. If the items are large, the benefits of the local search can be exploited and significantly better solutions are obtained by ALNS×LS_DBLF and ALNS×LS_OS compared to ALNS×DBLF. However, it can lead to larger computational requirements for the LS-based packing approaches for instances with large items. Although one iteration of the local search

is presumably faster, more iterations may be required for large items as finding a feasible packing pattern for them is more difficult. Thus, the lower requirements in computing times for the generation of one packing pattern is offset by needing more iterations for finding a feasible pattern. In contrast, the DBLF heuristic is aborted as soon as no feasible placement is found for any item and the difficulty of the packing problem does not impact the computing time in this way.
The third factor is the influence on the difficulty of the routing problem. With large items, finding a good and feasible solution is more complicated as it is more difficult to combine customers and to realize good insertions (with respect to the changes in the TTD). Thus, more iterations of the ALNS are required for obtaining a high-quality solution. This reflects in the large numbers of iterations that are conducted for instances with large items. In contrast, the best found solutions for instances with small items are reached quickly and the ALNS can be aborted earlier (due to the termination criterion of iterations without further improvements).

Number of items

The results separated by the total number of items for the 3L-VRPTW and rear and side loading are presented in Table 5.20 with the same performance indicators as in Table 5.19. The results are aggregated for the two loading approaches and divided by the item sizes due to their huge impact on the performance of the solution approaches. As before, only instances with $n = 60$ and $n = 100$ are included.
For small item instances, it can be observed that the relative solution quality of ALNS×LS_DBLF and ALNS×LS_OS deteriorates with an increasing number of items, i.e. the (positive) deviations from the benchmarks increase from 0.05 % (ALNS×LS_DBLF) and 0.91 % (ALNS×LS_OS) for $m = 200$ to 0.47 % and 1.93 % for $m = 400$. However, if the items are large, the relative solution quality of ALNS×LS_DBLF and ALNS×LS_OS improves with an increasing number of items, i.e. the negative deviations increase. For $m = 200$, they amount to -13.04 % (ALNS×LS_DBLF) and -13.21 % (ALNS×LS_OS). For $m = 400$, deviations of -18.82 % and -16.61 % are obtained. For both large and small items and all variants of the hybrid algorithm, the average computing times are longer, less iterations are conducted and more runs are terminated by the time limit if instances with 400 items are considered compared to instances with 200 items.

Table 5.20: Comparison of hybrid ALNS algorithms with different packing heuristics; extended problem variants (3L-VRPTW, RL/SL, C1); average TTDs, deviations from benchmark, computing times, numbers of iterations, proportions of prematurely aborted runs; separated by item size and total number of items m, restricted to $n \in \{60, 100\}$ and aggregated over both loading approaches

		ALNS ($n \in \{60, 100\}$)			
		small items		**large items**	
pack	**KPI**	$m = 200$	$m = 400$	$m = 200$	$m = 400$
DBLF	∅TTD*[DU]	964.64	981.64	1,630.18	2,926.37
	∅dev[%]	**0.00**	**0.00**	0.00	0.00
	∅ct[s]	1,290.27	1,968.57	111.60	169.39
	∅iter	13,759.02	11,327.97	21,025.57	23,554.92
	abort[%]	16.75	47.00	0.00	0.00
LS_DBLF	∅TTD[DU]	965.00	986.25	1,420.32	2,373.83
	∅dev[%]	0.05	0.47	−13.04	**−18.82**
	∅ct[s]	1,727.22	2,108.34	1,485.56	3,114.40
	∅iter	10,297.91	8,082.62	22,423.72	19,094.14
	abort[%]	40.00	50.25	14.00	65.79
LS_OS	∅TTD[DU]	972.00	999.09	1,417.79	2,436.11
	∅dev[%]	0.91	1.93	**−13.21**	−16.61
	∅ct[s]	1,934.24	2,927.30	2,799.04	3,396.08
	∅iter	7,890.81	6,460.60	18,115.74	11,230.84
	abort[%]	50.00	65.75	49.75	85.26

∅: average, *: benchmark, abort: share of runs aborted by time limit, ct: computing time per instance and run, dev: deviation from TTDs of benchmarks, DU: distance units, iter: total number of iterations, KPI: key performance indicator, m: number of items, pack: integrated packing heuristic, TTD: total travel distance.
Minimum average deviations per column are bold-faced.

Increasing the total number of items m makes the optimization problem, in particular the packing subproblem, more difficult to solve and more time-consuming. As different tendencies can be observed for the different item sizes, it can be assumed that an increasing total number of items intensifies the effects described above. On the one hand, the large computational requirements of instances with small items lead to a reduction of the total number of iterations conducted by the hybrid algorithms if the number of items is increased. As even less iterations can be performed within the time limit (which is reached more often), the results of ALNS×LS_DBLF and ALNS×LS_OS show larger deteriorations compared to the results of ALNS×-DBLF. On the other hand, if the items are large, the integrated ALNS is either finished within the computing time limit, i.e. the routing problem is solved as well as possible, or at least a sufficiently large of number of iterations can be conducted. In these cases, the benefits of the more sophisticated packing heuristics are exploited

stronger. Hence, it seems plausible that the (absolute) deviations between the hybrid algorithm with DBLF and the hybrid algorithms with the LS heuristics increase in any case for $m = 400$ compared to $m = 200$.

Number of customers

The results separated by the number of customers for the 3L-VRPTW and rear and side loading are presented in Table 5.21 with the same performance indicators as in Table 5.19. As before, the results are aggregated for both loading approaches and the solutions provided by ALNS×DBLF serve as benchmarks. In addition, the average numbers of iterations conducted per second (∅ips) are listed. The results are divided by the item sizes in order to include the instances with 20 customers into the analysis.

Table 5.21: Comparison of hybrid ALNS algorithms with different packing heuristics; extended problem variants (3L-VRPTW, RL/SL, C1); average TTDs, deviations from benchmark, computing times, numbers of iterations, proportions of prematurely aborted runs; separated by item size and number of customers n, and aggregated over both loading approaches

		ALNS				
		small items			**large items**	
pack	**KPI**	$n = 20$	$n = 60$	$n = 100$	$n = 60$	$n = 100$
DBLF	∅TTD[DU]	458.70	817.06	1,102.49	1,725.08	2,096.60
	∅dev[%]	0.00	0.00	**0.00**	0.00	0.00
	∅ct[s]	134.58	1,423.80	1,703.26	40.04	228.70
	∅iter	8,452.67	10,680.32	13,007.22	18,503.24	22,800.58
	∅ips	1,323.78	192.25	99.24	793.10	185.21
	abort[%]	2.50	26.50	36.92	0.00	0.00
LS_DBLF	∅TTD[DU]	445.28	816.10	1,107.47	1,487.85	1,864.05
	∅dev[%]	**−2.78**	**−0.11**	0.54	−13.20	**−9.72**
	∅ct[s]	546.94	1,702.89	1,752.02	1,031.49	2,279.29
	∅iter	4,060.00	7,837.81	11,584.44	20,733.98	19,311.98
	∅ips	890.33	186.92	95.79	185.20	35.66
	abort[%]	55.67	39.50	39.08	3.90	42.62
LS_OS	∅TTD[DU]	447.17	820.02	1,113.57	1,481.93	1,887.73
	∅dev[%]	−2.39	0.43	1.16	**−13.56**	−8.66
	∅ct[s]	550.24	1,964.80	2,143.48	1,771.97	2,896.76
	∅iter	3,636.09	6,996.42	10,265.78	19,351.79	14,682.24
	∅ips	590.48	62.57	35.12	107.91	14.74
	abort[%]	57.92	45.00	46.33	23.14	63.41

∅: average, *: benchmark, abort: share of runs aborted by time limit, ct: computing time per instance and run, dev: deviation from TTDs of benchmarks, DU: distance units, ips: number of iterations per second, iter: total number of iterations, KPI: key performance indicator, n: number of customers, pack: integrated packing heuristic, TTD: total travel distance.
Minimum average deviations per column are bold-faced.

The relative solution quality of ALNS×LS_DBLF and ALNS×LS_OS decreases with an increasing number of customers. For example, the deviations of the TTDs of the solutions obtained by ALNS×LS_DBLF from the TTDs of the benchmarks increase from -2.78 % ($n = 20$) to -0.11 % ($n = 60$) and 0.54 % ($n = 100$) in the case of small items. In the case of large item instances, the deviations amount to -13.2 % for $n = 60$ and to (only) -9.72 % for $n = 100$. Similar developments can be observed for ALNS×LS_OS. All three variants of the hybrid algorithm result in longer computing times if the number of customers increases. For example, approximately 9.1 minutes ($n = 20$), 28.4 minutes ($n = 60$) and 29.2 minutes ($n = 100$) are required by ALNS×LS_DBLF if instances with small items are considered. Moreover, the average numbers of conducted iterations increase with an growing number of customers if the items are small. This can be noticed for all three hybrid algorithms. In the case of large items, ALNS×DBLF performs more iterations for instances with 60 customers (18,503) than for instances with 100 customers (22,801). However, the opposite is true for ALNS×LS_DBLF and ALNS×LS_OS.

The number of customers affects both the routing and the packing subproblem. An increasing number of customers makes the underlying routing problem more difficult and more time-consuming to solve. It is more difficult because there are more possibilities to combine customers to routes. Therefore, more ALNS iterations are theoretically required for solving the problem. As noted above, this tendency can be observed for instances with small items. In this case, the shares of aborted runs do not vary much[42] and it can be assumed that the larger numbers of iterations results from the increasing difficulty of the routing problem. For large items, ALNS×LS_DBLF and ALNS×LS_OS conduct less iterations for instances with 100 than for those with 60 customers. However, here the larger computational efforts presumably reflect in the shares of runs aborted by the time limit, which increase significantly for ALNS×LS_DBLF and ALNS×LS_OS with a growing number of customers. It can be assumed that the lower numbers of iterations for $n = 100$ are caused by the comparatively large numbers of aborted runs. Furthermore, a single ALNS iteration requires more computation time. In particular, the insertion becomes very time-consuming with an increasing number of customers due to many possible insertions

[42] At least among $n = 60$ and $n = 100$. The comparatively large share for $n = 20$ shall be disregarded here. Note that the algorithms are aborted after 15 minutes if $n = 20$ and after 60 minutes for larger instances. It can be assumed that more runs are terminated prematurely due to the shorter time limit.

to be tested. This reflects in the numbers of iterations that can be conducted within one second. For example, they decrease from about 1,324 for $n = 20$ to ca. 192 for $n = 60$ and 99 for $n = 100$ if ALNS×DBLF and instances with small items are regarded.

Moreover, decreasing the number of customers leads to an increase in the number of items per customer as the total number of items is kept constant. An increasing number of items per customer appears to make the packing problem more difficult. As noted above, the more complex packing approaches are more beneficial for $m = 400$ than for $m = 200$. Similar trends can be observed here. That is, ALNS×LS_DBLF and ALNS×LS_OS are more beneficial (compared to ALNS×DBLF) for a lower number of customers.

Time window width

In Table 5.22, the results of the hybrid algorithms for the 3L-VRPTW with rear and side loading are separated by the time window width. The ALNS×DBLF solutions are the benchmark solutions. The previously introduced KPIs are employed and the results are aggregated for both loading approaches.

The average deviations from the benchmark TTDs obtained by ALNS×LS_DBLF and ALNS×LS_OS do not vary considerably for the different time window widths. The improvements compared to ALNS×DBLF for instances with narrow time windows are larger (average deviations for ALNS×LS_DBLF: −4.61 %, ALNS×LS_OS: −4.58 %) than for instances with wide time windows (ALNS×LS_DBLF: −4.22 %, ALNS×LS_OS: −3.24 %). For all three variants of the hybrid algorithm, solving instances with wide time windows requires significantly more computing time, less iterations are conducted per second and more runs are terminated by the time limit. The time window widths restrict the routing problem. Narrow time windows lead to less possibilities to form routes, which – as a result – are shorter than routes for instances with wide time windows. Shorter routes contain fewer items to be packed. Consequently, the computational effort of solving one packing problem is lower. Furthermore, less possibilities to form routes result in less possible insertions (during the execution of an insertion heuristic) and, thus, in less calls of the packing procedures. Therefore, the computing times for the instance classes with narrow time windows are considerably lower than for instances with wide time windows.

The quality of ALNS×LS_DBLF and ALNS×LS_OS compared to ALNS×DBLF is

slightly worse for instances with wide time windows than for those with narrow time windows. This is presumably caused by the increasing computational effort and the fact that less iterations can be conducted within the time limit. The average total numbers of conducted iterations are similar for instances with narrow time windows. For instances with wide time windows, though, significantly fewer iterations are conducted by ALNS×LS_DBLF (8,805) and ALNS×LS_OS (6,278) than by ALNS×-DBLF (13,022). Problem classes for which the best results are obtained by the DBLF heuristic are almost exclusively classes with wide time windows.

Table 5.22: Comparison of hybrid ALNS algorithms with different packing heuristics; extended problem variants (3L-VRPTW, RL/SL, C1); average TTDs, deviations from benchmark, computing times, numbers of iterations, proportions of prematurely aborted runs; separated by time window width and aggregated for both loading approaches

		ALNS	
pack	**KPI**	**narrow time windows**	**wide time windows**
DBLF	∅TTD*[DU]	1,284.01	1,108.34
	∅dev[%]	0.00	0.00
	∅ct[s]	108.48	1,408.19
	∅iter	15,641.73	13,021.84
	∅ips	869.25	136.01
	abort[%]	0.00	28.50
LS_DBLF	∅TTD[DU]	1,193.42	1,030.63
	∅dev[%]	**−4.61**	**−4.22**
	∅ct[s]	559.71	2,408.80
	∅iter	15,313.28	8,804.76
	∅ips	549.77	27.24
	abort[%]	11.28	65.51
LS_OS	∅TTD[DU]	1,193.14	1,044.13
	∅dev[%]	−4.58	−3.24
	∅ct[s]	1,061.53	2,659.68
	∅iter	14,412.99	6,278.15
	∅ips	309.29	26.40
	abort[%]	19.17	78.23

∅: average, *: benchmark, abort: share of runs aborted by time limit, ct: computing time per instance and run, dev: deviation from TTDs of benchmarks, DU: distance units, ips: number of iterations per second, iter: total number of iterations, KPI: key performance indicator, pack: integrated packing heuristic, TTD: total travel distance. Minimum average deviations per column are bold-faced.

Number of item types

The heterogeneity of the items is defined by the number of different item types. The respective results for the 3L-VRPTW and rear and side loading are presented in Table 5.23. As before, the solutions provided by ALNS×DBLF serve as benchmarks.

The same performance indicators as above are used and the results are aggregated for the two loading approaches.

Table 5.23: Comparison of hybrid ALNS algorithms with different packing heuristics; extended problem variants (3L-VRPTW, RL/SL, C1); average TTDs, deviations from benchmark, computing times, numbers of iterations, proportions of prematurely aborted runs; separated by the number of item types and aggregated for both loading approaches

		ALNS		
pack	**KPI**	**3 types**	**10 types**	**100 types**
DBLF	∅TTD*[DU]	1,194.97	1,176.92	1,216.51
	∅dev[%]	0.00	0.00	0.00
	∅ct[s]	672.31	818.78	793.66
	∅iter	14,027.19	14,244.65	14,742.09
	abort[%]	12.66	16.03	14.25
LS_DBLF	∅TTD[DU]	1,142.19	1,084.24	1,107.13
	∅dev[%]	**−2.69**	**−4.89**	**−5.79**
	∅ct[s]	1,247.17	1,555.55	1,673.43
	∅iter	12,404.33	11,897.63	11,831.21
	abort[%]	28.22	42.09	45.81
LS_OS	∅TTD[DU]	1,143.25	1,092.01	1,118.60
	∅dev[%]	−2.67	−4.29	−4.87
	∅ct[s]	1,354.73	2,030.14	2,240.01
	∅iter	12,367.86	9,637.27	8,854.94
	abort[%]	28.11	55.81	63.91

∅: average, *: benchmark, abort: share of runs aborted by time limit, ct: computing time per instance and run, dev: deviation from TTDs of benchmarks, DU: distance units, iter: total number of iterations, KPI: key performance indicator, pack: integrated packing heuristic, TTD: total travel distance.
Minimum average deviations per column are bold-faced.

With an increasing number of item types, the (absolute) deviations from the benchmark TTDs, i.e. the improvements, obtained by ALNS×LS_DBLF and ALNS×LS_OS increase. For ALNS×LS_DBLF (ALNS×LS_OS), the average deviations amount to −2.69 % (−2.67 %) for instances with three types, −4.89 % (−4.29 %) for instances with ten types and −5.79 % (4.87 %) for instances with 100 different item types. A larger number of item types has the following effects: The average computing times increase for all variants of the hybrid algorithm. Moreover, the average total numbers of iterations decrease and the shares of aborted runs increase in the case of ALNS×LS_DBLF and ALNS×LS_OS. For ALNS×DBLF, the number of iterations increases and no clear trend can be observed with respect to the shares of aborted runs.

Instances with strongly heterogeneous item sets contain more difficult packing problems than instances with weakly heterogeneous item sets. This assumption is also

supported by the results of the hybrid algorithm. The computing times usually increase with a larger number of item types. There are exceptions, though, as the computing times also depend heavily on other instance characteristics. In addition, the (absolute) deviations between the TTDs of the benchmarks and of the solutions of ALNS×LS_DBLF and ALNS×LS_OS increase with an increasing number of item types. The increasing difficulty of the packing problem can apparently be handled better by the more complex packing heuristics.

Linehaul share

The linehaul share[43] can impact how beneficial different loading approaches are – in particular for transporting linehaul and backhaul items simultaneously. Therefore, the focus here will not be on the comparison of hybrid algorithms but on the comparison of the loading approaches. Moreover, this analysis concentrates on the 3L-VRPMB(TW) and 3L-VRPSDP(TW). These problem variants are affected more strongly by varying linehaul shares than the 3L-VRPCB(TW) due to the simultaneous transport of linehaul and backhaul items.

The results for the comparison of the loading approaches LSP and side loading for 3L-VRPMB(TW) and 3L-VRPSDP(TW) are presented in Table 5.24. The method of analysis is equivalent to the one used for Tables 5.11, 5.14 and 5.16. That is, pairs of instances are compared that differ only in the loading approach. The solutions of the LSP instances are taken as benchmarks for the individual hybrid algorithms. The results in Table 5.24 are averaged over all regarded variants of the hybrid algorithm. The average TTDs (⌀TTD) of the instances with the different loading approaches, the resulting average deviations (⌀dev) from the benchmark TTDs and the average numbers of used vehicles ($\varnothing v_{used}$) are presented. The results are aggregated over the problem variants with and without time windows.

On average, the side loading approach is more beneficial than the LSP approach, which was noted above for the individual problem variants. If half of the customers are linehaul customers in the 3L-VRPMB(TW), the average deviations of the TTDs of the side loading solutions from the TTDs of the LSP solutions amount to −8.86 %. They amount to −11.61 % for a linehaul share of 80 %. In the case of the 3L-VRPSDP(TW), the TTDs of the side loading solutions deviate on average −4.01 %

[43] It represents the share of linehaul customers in the case of 3L-VRPCB(TW) and 3L-VRPMB(TW) and the share of linehaul items of all items of each customers in the case of the 3L-VRPSDP(TW) (see Chapter 5.2.2).

(linehaul share of 50 %) and −11.03 % (linehaul share of 80 %) from the benchmark TTDs. Moreover, less vehicles are required if side loading is applied.

Table 5.24: Comparison of loading approaches; extended problem variants (3L-VRPMB(TW), LSP/SL, C1) and (3L-VRPSDP(TW), LSP/SL, C1); average TTDs, deviations from benchmark, numbers of used vehicles; separated by problem variant and linehaul share, aggregated over the variants of the hybrid algorithm

		ALNS			
		3L-VRPMB(TW)		**3L-VRPSDP(TW)**	
load	**KPI**	50 % LH	80 % LH	50 % LH	80 % LH
LSP	∅TTD*[DU]	877.99	1,023.05	1,006.49	1,124.24
	∅dev[%]	0.00	0.00	0.00	0.00
	$\varnothing v_{used}$	7.99	11.08	10.63	12.79
SL	∅TTD[DU]	764.71	830.27	937.87	917.90
	∅dev[%]	**−8.86**	**−11.61**	**−4.01**	**−11.03**
	$\varnothing v_{used}$	5.84	7.33	9.06	8.89

∅: average, *: benchmark, dev: deviation from TTDs of benchmarks, DU: distance units, KPI: key performance indicator, load: applied loading approach, LSP: loading space partition, SL: side loading, TTD: total travel distance, v_{used}: number of used vehicles.
Minimum average deviations per column are bold-faced.

It is assumed here that the loading space is separated into two sections of equal size in the case of the LSP approach. Therefore, this approach is more beneficial for instances with equal shares of linehaul and backhaul customers or items, respectively. As a result, the benefits of the side loading approach (over the LSP), i.e. the absolute deviations from the benchmark TTDs, increase with an increasing linehaul share.
The results further suggest that, compared to a 3L-VRPSDP(TW), the benefits of the side loading approach (utilization of the whole loading space, simplicity of arranging items) are exploited stronger if the 3L-VRPMB(TW) is considered. As mentioned above, this is due to the different problem characteristics. As every route of a 3L-VRPSDP(TW) solution must contain both linehaul and backhaul items, the LSP appears to be a reasonable approach for the 3L-VRPSDP(TW) – in particular, if the shares of linehaul and backhaul items are equal. If 80 % of the customers or items are linehauls, both the 3L-VRPMB(TW) and the 3L-VRPSDP(TW) solutions show similar deviations of the TTDs of the side loading solutions from those of the LSP solutions. In this case, it can be assumed that considerably more linehaul items than backhaul items are transported on every route. For the 3L-VRPMB(TW) it is even likely that routes are formed without any customer of one type (linehaul or backhaul). In Table 5.25, the shares of routes containing both linehaul and backhaul customers (mix) in the solutions of the 3L-VRPMB(TW) are listed for the linehaul

shares of 50 % and 80 %. The numbers are aggregated over the variants of the hybrid algorithm. Whereas almost all routes (96.7 % over both loading approaches) contain both types of customers in the case of a linehaul share of 50 %, this is true for only 72.9 % (over both loading approaches) if 80 % of the customers are linehaul customers. If only linehaul customers or only backhaul customers are served in a route, half of the loading space remains unused. Hence, the LSP appears to be an inefficient approach in this case. Using the side loading approach, one can react more flexibly to varying linehaul shares.

Table 5.25: Shares of routes with both linehaul and backhaul customers in solutions of the extended problem variants (3L-VRPMB(TW), LSP/SL,C1); separated by linehaul share and aggregated over the variants of the hybrid algorithm

		ALNS	
load	**KPI**	**50 % LH**	**80 % LH**
LSP	mix[%]	97.1	67.7
SL	mix[%]	96.3	78.1
total	mix[%]	96.7	72.9

load: applied loading approach, KPI: key performance indicator, LH: linehaul, LSP: loading space partition, mix: share of routes containing both linehaul and backhaul customers, SL: side loading

5.5.2.3 Comparison of backhaul variants

In the following, the benefits of different variants of incorporating backhauls into the routing problem are compared and evaluated. However, only the backhaul variants "clustered backhauls" and "mixed backhauls" are considered. The backhaul variant "simultaneous pickup and delivery" is not regarded, as the different problem characteristics (division of items instead of customers into linehaul and backhaul sets) do not allow for an obvious comparison.

In order to carry out a fair and meaningful evaluation of the backhaul variants "clustered backhauls" and "mixed backhauls", the following procedure is chosen:

(1) All core instances of the problem variants 3L-VRPCB(TW) and 3L-VRPMB-(TW) are regarded (for a restriction regarding the number of items see below). A comparable pair of such core instances is given if the related basic instances are identical, both core instances have the same time window specification (i.e. time windows are specified or not) and, thus, they differ only in the backhaul variant.

(2) For each core instance of a comparable pair, both related instances with an appropriate loading approach (rear and side loading for clustered backhauls, LSP and side loading for mixed backhauls) and constraint set C1 are considered. Each of the two instances is solved five times by each of the three variants of the hybrid algorithm. The best resulting solution (over both instances) with minimum TTD is called the best solution of the core instance. The best solutions of the corresponding core instances of all comparable pairs are then compared (i.e. the deviations of the TTDs are calculated) where the 3L-VRPCB(TW) solutions serve as benchmarks.

(3) In addition, the related instances of a core instance of a comparable pair are also solved by transporting linehaul and backhaul items strictly in separate routes. This additional problem variant is also considered with and without time windows and the loading approaches rear and side loading (as implemented for the 3L-VRPTW and 3L-VRPCB(TW)) are applied.

The results are given in Table 5.26. They are presented separately for the different item sizes and include only instances with 200 items, since those with 400 items are not considered for the 3L-VRPMB(TW). The additional problem variant of serving linehaul and backhaul customers in individual tours is denoted as "separate". The average TTDs (∅TTD) of the best solutions of the core instances of each considered problem variant and their deviations (∅dev) from the TTDs of the benchmark solutions (best solutions of the corresponding 3L-VRPCB(TW) core instances) are presented. As this analysis focuses on the solution quality, an analysis of the computational efforts is neglected. The results are aggregated for the problem variants with and without time windows.

If the items are small, the 3L-VRPMB(TW) solutions are clearly better than the 3L-VRPCB(TW) solutions with an average benchmark deviation of −16.69 %. If the items are large, however, the solutions of the 3L-VRPCB(TW) are slightly better. In this case, the TTDs of the 3L-VRPMB(TW) deviate on average 0.21 % from the benchmark TTDs. In both cases, serving linehaul and backhaul customers strictly in separate routes leads to considerably worse solutions than integrating backhaul transportations into delivery tours. The TTDs of the corresponding solutions deviate on average 10.78 % (small items) and 27.07 % (large items) from the TTDs of the 3L-VRPCB(TW) solutions. Considering small items, more customers can be

merged in one route resulting in more solution possibilities. Thus, the drawbacks of separate tours for linehaul and backhaul customers can partially be compensated by generating low-cost routes with comparatively large numbers of customers. In contrast, the possibilities of adding customers to a route and sequencing them efficiently are very limited if the transported items are large.

Table 5.26: Comparison of backhaul problem variants; extended problem variants (3L-VRPCB(TW), RL/SL, C1), (3L-VRPMB(TW), LSP/SL, C1) and (separate, RL/SL, C1); average TTDs and deviations from benchmarks; separated by item size and limited to $m = 200$

		ALNS ($m = 200$)		
problem	**KPI**	**small items**	**large items**	**total**
3L-VRPCB(TW)	∅TTD*[DU]	772.17	1,188.06	938.53
	∅dev[%]	0.00	**0.00**	0.00
3L-VRPMB(TW)	∅TTD[DU]	637.72	1,186.18	857.11
	∅dev[%]	**−16.69**	0.21	**−9.93**
separate	∅TTD[DU]	840.79	1,489.08	1,100.10
	∅dev[%]	10.78	27.07	17.30

∅: average, *: benchmark, dev: deviation from TTDs of benchmarks, DU: distance units, KPI: key performance indicator, TTD: total travel distance. Minimum average deviations per column are bold-faced.

The results comparing the 3L-VRPCB(TW) to the 3L-VRPMB(TW) are slightly ambiguous with considerable differences in the benchmark deviations for the different item sizes. A priori, it could have been expected that visiting linehaul and backhaul customers in arbitrary sequences (3L-VRPMB(TW)) is more beneficial than clustering linehauls and backhauls (3L-VRPCB(TW)) due to less restrictions in forming the routes. The experiments on instances with small items confirm these expectations. However, the instances with large items show an opposite result. The simultaneous transport of linehaul and backhaul items appears to be crucial for those deviations. Arranging small items in a way that linehaul and backhaul items do not block each other, is not as difficult as arranging large items. Thus, the advantages of the routing with mixed backhauls can be exploited to a greater extent. In contrast, the solutions of the two problem variants seem similar for large items, suggesting that the possibilities of mixing linehaul and backhaul customers in the routing sequence are limited.

In Table 5.27, the average proportions of all customers (custBH) and of the linehaul customers (LHBH) of a route that are visited before the first backhaul customer is approached, are presented for instances with small and large items. For this purpose,

only routes containing both linehaul *and* backhaul customers are considered. Moreover, the average shares of routes with both linehaul and backhaul customers are listed (mix). These results are summarized for all variants of the hybrid algorithm.

Table 5.27: Average shares of linehaul or all customers of a route visited before the first backhaul customer and proportion of routes with both linehaul and backhaul customers; extended problem variants (3L-VRPCB(TW), RL/SL, C1) and (3L-VRPMB(TW), LSP/SL, C1); separated by item size, limited to $m = 200$, aggregated over all variants of the hybrid algorithm and loading approaches

ALNS ($m = 200$)				
problem	**KPI**	**small items**	**large items**	**total**
3L-VRPCB(TW)	∅LHBH [%]	100.0	100.0	100.0
	∅custBH [%]	60.6	59.8	60.3
	∅mix [%]	79.5	73.1	76.9
3L-VRPMB(TW)	∅LHBH [%]	27.4	60.8	40.8
	∅custBH [%]	18.9	37.1	26.2
	∅mix [%]	91.9	74.2	84.8

∅: average, custBH: proportion of customers of the route visited before the first backhaul customer, KPI: key performance indicator, LHBH: proportion of linehaul customers of the route visited before the first backhaul customer, mix: proportion of routes containing both linehaul and backhaul customers

On average, about 61 % of the linehaul customers of a route are visited before the first backhaul customer in the solutions of the 3L-VRPMB(TW) and instances with large items. Moreover, the first backhaul customer is approached after, on average, about 37 % of all customers in the respective routes have been visited. Due to the properties of the problem, the first backhaul customer is approached later in the 3L-VRPCB(TW). Nearly 60 % of the route is executed before the first backhaul customer is serviced. In contrast, only about 27 % of a route's linehaul customers and about 19 % of all customers of a route are visited before the first backhaul customer if the items are small in the 3L-VRPMB(TW). Furthermore, linehaul *and* backhaul customers are serviced in about 92 % of the routes of the 3L-VRPMB(TW) solutions for instances with small items, whereas only 74 % of the routes in solutions for instances with large items contain both types of customers. Thus, large items restrict the possibilities to mix linehaul and backhaul customers in a route. In this case, clustering the customers like in the 3L-VRPCB(TW) can even be more beneficial as the items can be arranged more efficiently in the loading space.

5.5.2.4 Influence of time windows

In the following, the impact of having to consider time windows is evaluated. For this

purpose, pairs of core instances are compared that differ exclusively in the presence or absence of time windows. That is, core instances are matched that refer to the same backhaul variant (core instances of the 3L-VRPCB are compared to those of the 3L-VRPCBTW, etc.). All 3L-VRPBTW variants are taken into consideration apart from the 3L-VRPTW as this problem variant is not considered without time windows. The results are presented in Table 5.28. As in Chapter 5.5.2.3, the best solutions of the core instances are subject of the analysis. The best solutions of the core instances without time windows serve as benchmarks.

In Table 5.28, the average TTDs (∅TTD) of the best solutions, their average deviations (∅dev) from the benchmark TTDs, the average numbers of used vehicles ($\varnothing v_{used}$) and the average computing times (∅ct) per instance and run are given. The results are summarized for the considered 3L-VRPBTW variants and presented for instances with narrow and wide time windows separately and altogether.

Table 5.28: Comparison of results obtained with and without consideration of time windows; extended problem variants (3L-VRPCB(TW), RL/SL, C1), (3L-VRPMB(TW), LSP/SL, C1), (3L-VRPSDP(TW), LSP/SL, C1); average TTDs, deviations from benchmarks, numbers of used vehicles, computing times; separated by time window width and aggregated over the problem variants

		ALNS		
time windows	**KPI**	**narrow time windows**	**wide time windows**	**total**
no	∅TTD*[DU]	868.13	859.77	863.95
	∅dev[%]	**0.00**	**0.00**	**0.00**
	$\varnothing v_{used}$	7.67	7.53	7.60
	∅ct[s]	2,101.94	2,142.20	2,122.07
yes	∅TTD[DU]	1,181.40	985.23	1,083.32
	∅dev[%]	43.91	17.44	30.67
	$\varnothing v_{used}$	12.98	8.95	10.96
	∅ct[s]	313.07	1,539.26	926.17

∅: average, *: benchmark, ct: computing time per instance and run, dev: deviation from TTDs of benchmarks, DU: distance units, TTD: total travel distance, v_{used}: number of used vehicles.
Minimum average deviations per column are bold-faced.

Considering time windows costs an average deterioration of the TTDs of 30.67 %. As narrow time windows are more restrictive, they lead to larger deviations from the benchmarks without time windows (43.91 %) compared to instances with wide time windows (17.44 %). If time windows are neglected, on average 7.6 vehicles are used. Otherwise, 8.95 vehicles (wide time windows) and 12.98 vehicles (narrow time windows) are needed on average. Furthermore, the computing times for instances

without time windows (2,122 seconds) are noticeably longer than those for instances with time windows (926 seconds). Among the latter, it can be observed that longer computing times are required for solving an instance with wide time windows (1,539 seconds) compared to instances with narrow time windows (313 seconds).
The results suggest that the consideration of time windows becomes more costly the more restrictive the time windows are. Low-cost routes that may be realized if time windows are not considered might not be possible if they led to a violation of the time windows. Furthermore, the more restrictive the time windows are, the less customers can be merged to routes. As a consequence, more vehicles are needed, which also reflects in higher routing costs.
Narrow time windows make the routing subproblem more difficult to solve but less time-consuming, as they allow for fewer possibilities to form routes. Consequently, there are fewer options to insert customers into existing routes in the course of the application of an insertion heuristic. The adherence to the time windows is checked before a route is checked for packing feasibility. Thus, many routes are identified as infeasible without calling a packing procedure, which leads to lower computational efforts.

5.5.2.5 Influence of three-dimensional loading

How sensible is it to solve a VRP under consideration of three-dimensional loading constraints? Might it be sufficient to relax the loading constraints and solve the problem solely by taking (one-dimensional) capacity constraints into account? In order to answer these questions, the 3L-VRPBTW instances are solved as one-dimensional instances.[44] That is, instead of generating packing plans for the routes, only the weight and volume restrictions of the vehicles are regarded. All seven problem variants and the related loading approaches are taken into account. The results are presented in Table 5.29. If the instances are solved considering 1D capacity constraints, the minimum TTD solutions among five runs are regarded for each instance. If 3D loading constraints are considered, the best solutions per instance among all runs, loading approaches and variants of the hybrid algorithm are regarded. The 1D solutions serve as benchmarks. The average TTDs (∅TTD), their average deviations (∅dev) from the benchmark TTDs and the average computing

[44] Hereinafter, the solutions of the one-dimensional VRPBTWs are also referred to as "1D solutions". Analogously, the solutions of 3L-VRPBTWs are called "3D solutions".

times (∅ct) are given in Table 5.29. As before, the results are presented for instances with small and large items separately and the impact of the number of customers is excluded by neglecting instances with 20 customers.

Table 5.29: Comparison of VRP solutions without loading constraints (1D) and with 3D loading constraints; extended problem variants (3L-VRPTW/3L-VRPCB(TW), RL/SL, C1), (3L-VRPMB(TW)/3L-VRPSDP(TW), LSP/SL, C1); average TTDs and computing times, average deviations from benchmarks (1D); separated by item sizes and restricted to $n \in \{60, 100\}$

ALNS ($n \in \{60, 100\}$)				
constraints	**KPI**	**small items**	**large items**	**total**
1D	∅TTD*[DU]	873.12	1,228.39	1,050.76
	∅dev[%]	**0.00**	**0.00**	**0.00**
	∅ct[s]	15.20	11.94	13.57
3D	∅TTD[DU]	880.76	1,452.66	1,166.71
	∅dev[%]	1.27	20.82	11.04
	∅ct[s]	2,104.73	1,340.96	1,722.84

∅: average, *: benchmark, ct: computing time per instance and run, dev: deviation from TTDs of benchmarks, DU: distance units, KPI: key performance indicator, TTD: total travel distance.
Minimum average deviations per column are bold-faced.

The consideration of three-dimensional loading constraints leads to solutions with TTDs that are on average 11.04 % longer than those obtained by neglecting the loading constraints. This suggests that the solutions differ considerably and that the solutions obtained without taking the packing problem into account might not be feasible if they had to be executed. In addition, the computing times differ extremely. While solving a (one-dimensional) VRPBTW requires on average only about 14 seconds, approximately 29 minutes are needed for solving a 3L-VRPBTW. Those differences indicate that about 99 % of the computing time of the hybrid algorithm is spent on the packing procedures.

The item sizes have a huge impact on the discrepancies between the 1D and 3D solutions. As mentioned before, the instances with small items might not be solved as effectively as possible with respect to the routing subproblem because the hybrid algorithm is terminated by the computing time limit before reaching a sufficient number of ALNS iterations. The computing times for solving the 1D problem are comparatively small and a sufficient number of ALNS iterations can be conducted in order to obtain high-quality solutions. Hence, it could be assumed a priori that the 1D solutions differ significantly from the 3D solutions. However, the differences are low. The average deviations of the TTDs of the 3D solutions from those of

the 1D solutions amount to only 1.27 % for instances with small items. These minor differences indicate, that the hybrid algorithm for the 3L-VRPBTW variants succeeds in finding good solutions also for problem instances with small items – despite their large requirement in computation time.

The differences are significant, though, if large items are considered. The average deviations from the TTDs of the 1D solutions amount to 20.82 %. Although the volume (and weight) of a set of items might not exceed the capacity of a loading space, finding a feasible packing plan is not guaranteed. A set of small items is easier to pack than a set of large items with the same total volume as more possibilities of arranging the items exist. Hence, it can be expected that many routes are not executable if the items to be transported are large and 3D loading constraints are not considered in the planning process.

As smaller items are easier to pack, it is likely that the solutions obtained with and without consideration of loading constraints are similar. In conclusion, these instances (and comparable practical cases) could indeed be relaxed to 1D problems and solved as such within a few seconds. Slight modifications might be necessary in order to obtain a feasible solution for the 3D problem. However, solving the problem as an integrated routing and packing problem has the advantage that a packing plan, which observes the relevant packing constraints, is automatically provided. Ensuring these constraints manually can be a very time-consuming and difficult task. Furthermore, a packing plan might be required if the loading process is conducted automatically. On the other hand, the packing problem should certainly be considered if the transported items are large in order to ensure feasibility.

5.5.3 Consideration of additional packing constraints

Additional numerical experiments are conducted in order to analyse the influence of further packing restrictions. The constraints *reachability*, *robust stability* and *load bearing strength* have been incorporated into the packing construction heuristics (excluding the open space heuristic) as described in Chapter 4.1.6. Different sets of constraints are regarded, which are listed in Table 5.30[45]. C1 refers to the original set of constraints based on Gendreau et al. (2006), which is used in the previous experiments. The geometrical, vertical orientation and LIFO constraints are included in all sets. In the sets C2, C3 and C4, each of the new constraints is

[45] It resembles Table 2.1 and is placed here for the sake of convenience.

added individually. The vertical stability and fragility constraints are replaced by the robust stability and load bearing strength constraints, respectively. Therefore, these original constraints are excluded in the corresponding sets. Finally, set C5 contains all of the additional constraints.

Table 5.30: Overview of the sets of packing constraints

	packing constraints							
set	GC	VO	LIFO	VS	FR	R	RS	LB
C1	✓	✓	✓	✓	✓			
C2	✓	✓	✓	✓	✓	✓		
C3	✓	✓	✓		✓		✓	
C4	✓	✓	✓	✓				✓
C5	✓	✓	✓			✓	✓	✓

FR: fragility, GC: geometrical constraints, LB: load bearing strength, R: reachability, RS: robust stability, VO: vertical orientation, VS: vertical stability

In the following, only instances with large items are considered because the effects of the different packing constraints are most visible for these instances.

5.5.3.1 Randomly generated routes

The first experiments with the additional packing constraints consist in finding feasible packing plans for randomly generated routes in order to assess how restrictive the constraints are – in particular, in comparison to the original constraint set C1. The same routes as in previous experiments are used (cf. Chapter 5.5.1.1). About 12,000 routes are available for the instances with large items. The results are summarized in Table 5.31 and subdivided by intervals of volume utilization of the item sets corresponding to the routes. The table contains the average shares of feasibly packed random routes obtained by the heuristics DBLF and LS_DBLF[46] with the different constraint sets. In addition, the average computing times (⌀ct) in milliseconds per route are given. Bold-faced values indicate the constraint set with the largest share of packed routes per packing heuristic and interval of volume utilization.

The results confirm the assumption that the additional constraints restrict the packing problem further than the original constraints. Applying DBLF and C1, feasible packing plans are found for 61.4 % of all routes, while this is the case for only 57.7 % (C2), 61.0 % (C3), 59.0 % (C4) and 55.6 % (C5) when regarding the new constraint

[46] LS_OS is not regarded as the construction heuristic is not equipped with the additional constraints.

sets. Applying the LS_DBLF heuristic and C1, feasible packing plans are found for 77.4 % of all routes. Considering the new constraint sets, 72.4 % (C2), 79.9 % (C3), 76.0 % (C4) and 70.8 % (C5) of the routes can be packed feasibly by LS_DBLF. The set C3 (robust stability) seems to have the lowest impact, as only slightly fewer routes can be packed by applying C3 than by applying C1. However, the computing times are more than 60 times higher than the computing times with C1. Solving the packing problem for single routes only, this does not constitute a big impact as the computing times are still extremely low (DBLF: ca. 1 ms, LS_DBLF: ca. 14 ms). However, tens of thousands of packing problems need to be solved within the ALNS framework. It can be expected that the comparatively large requirements in computation times will have a greater effect on the hybrid algorithm. Among the constraint sets that only add one new constraint (C2-C4), the set C2, i.e. the application of the reachability constraint, leads to the lowest shares of feasibly packed routes. Thus, it appears to be the most restrictive constraint. Its application is not connected with considerable additional computational efforts since the computing times are marginally longer compared to C1. Combining all of the additional packing constraints (C5), feasible packing plans could be generated for the lowest number of routes and the computing times are about 141 (225) times longer than those needed with C1 when DBLF (LS_DBLF) is utilized.

Table 5.31: Comparison of constraint sets; shares of feasibly packed random routes and average computing times obtained by DBLF and LS_DBLF

		shares of feasibly packed routes [%] per utilization interval						**⌀ct**
heuristic	**set**	10-20	20-30	30-40	40-50	50-60	total	[ms]
DBLF	C1	**98.6**	**88.5**	**66.9**	**37.6**	**15.4**	**61.4**	0.014
	C2	98.4	87.2	61.4	31.0	10.7	57.7	0.058
	C3	**98.6**	**88.5**	66.5	36.6	14.7	61.0	0.904
	C4	**98.6**	88.0	65.1	32.8	10.3	59.0	0.095
	C5	98.3	86.4	59.6	26.5	7.1	55.6	2.011
LS_DBLF	C1	**100.0**	98.5	**91.0**	64.3	**33.2**	**77.4**	0.175
	C2	**100.0**	98.2	86.5	54.1	23.2	72.4	0.482
	C3	**100.0**	**98.7**	89.5	**64.9**	31.4	76.9	14.137
	C4	99.9	98.5	**91.0**	63.3	27.5	76.0	1.317
	C5	99.9	98.2	86.0	52.5	17.5	70.8	39.468

⌀ct: average computing time per route.
Intervals represent the minimum and maximum volume utilizations of the loading space. Maximum shares per column and heuristic are bold-faced.

For further evaluation whether it is reasonable to integrate extended packing con-

straints, the packing plans that are feasible with respect to C1 are tested for feasibility concerning the additional constraints. The results are presented in Table 5.32, listing the percentage shares of feasible C1 packing plans that are feasible regarding the reachability (reach), robust stability (rob) and load bearing strength (lbs) constraints. The results are subdivided by volume utilization intervals (vol) and packing heuristics.

Table 5.32: Shares [%] of feasible packing plans (C1) that are feasible w.r.t. the additional packing constraints; separated by volume utilization interval and packing heuristic

	shares [%] of feasible C1 packing plans								
	DBLF			**LS_DBLF**			**total**		
vol[%]	reach	rob	lbs	reach	rob	lbs	reach	rob	lbs
10-20	74.4	99.6	95.7	74.4	99.6	95.8	74.4	99.6	95.7
20-30	48.3	99.0	86.2	48.1	99.0	86.5	48.2	99.0	86.4
30-40	31.5	98.5	78.4	30.7	98.5	78.4	31.1	98.5	78.4
40-50	19.0	97.1	70.0	16.8	97.2	69.6	17.8	97.2	69.8
50-60	21.6	98.2	50.3	12.9	96.9	60.4	16.2	97.4	56.6
total	41.5	98.6	79.2	37.5	98.3	78.9	39.4	98.4	79.0

lbs: load bearing strength, reach: reachability, rob: robust stability, vol: volume utilization interval

Considerable differences can be observed between the constraints. The robust stability requirements are almost always fulfilled. 98.4 % of the packing plans that are feasible under consideration of C1 satisfy them, whereas the reachability (load bearing strength) constraint is only satisfied by 39.5 % (79.0 %) of the C1 packing plans. The shares of feasible packing plans decrease with increasing volume utilization. With respect to the robust stability constraint, the reductions are very low. For both packing heuristics combined, these shares fall from 99.6 % (10-20 % volume utilization) to 97.4 % (50-60 %). They decrease extremely if the adherence to the reachability constraint is tested (from 74.4 % to 16.2 %). In accordance with these findings, the proportions of routes with feasible packing plans considering C1 and C2 (see Table 5.31) increasingly diverge with increasing volume utilization rates.
The presented results indicate that the reachability and load bearing strength constraints should be integrated into a packing heuristic if they are of practical relevance. It is highly likely that otherwise generated packing plans (i.e. without taking the constraints into account) would violate these constraints. As opposed to this, it is to be expected that robust stability is provided in most cases by a packing plan that is generated without explicitly considering the constraint.

5.5.3.2 Hybrid algorithm

Furthermore, the hybrid ALNS algorithm including the packing heuristics DBLF and LS_DBLF is applied five times to the 3L-VRPBTW instances (with large items). As before, the feasible solution with minimum TTD (among the feasible solutions obtained in the five runs by a certain variant of the hybrid algorithm) is considered as the solution of the corresponding instance provided by the algorithm variant. All seven problem variants, related loading approaches and the five sets of packing constraints, as described above, are considered. That is, all 70 different extended problem variants are taken into account.

In the following, solutions of instances that are derived from the same core instances and that differ exclusively in the applied constraint set are compared. The solutions of the instances with the constraint set C1 serve as benchmarks. For example, the solutions of (3L-VRPTW, RL, C2/C3/C4/C5) instances are compared to the solution of the corresponding (3L-VRPTW, RL, C1) instance. Table 5.33 contains the average TTDs (⌀TTD) and the average and maximum deviations from the benchmark TTDs (⌀dev, max dev) obtained by applying the additional constraint sets. Moreover, the average deviations between the TTDs from those obtained by ALNS×DBLF are stated (devDBLF) as well as the average computing times per instance and run (⌀ct) and numbers of conducted iterations (⌀iter). The results are summarized over the instances of all problem variants and loading approaches, and provided for the approaches ALNS×DBLF and ALNS×LS_DBLF.

On average, the TTDs deviate 6.37 % (C2), 7.38 % (C3), 5.25 % (C4) and 11.43 % (C5) from the benchmark (C1) TTDs. Those deviations affirm that the packing subproblem becomes more restrictive due to the integration of the additional constraints. The integration of all restrictions (C5) causes the largest deviations from the benchmark solutions as the packing problems are most restrictive in this case. For individual instances, the TTDs even deviate up to 71.5 % from the benchmarks. However, not only the restrictiveness of the packing problem leads to large deviations from the benchmarks, but also the increased computational requirements. Increasing computing times result in the conduction of considerably fewer iterations within a given time limit. For example, the application of C3 (robust stability) leads to the second largest deviations from the C1 solutions among the sets C2-C5. Yet, previous experiments indicate that almost all solutions obtained under consideration

of C1 also satisfy the robust stability constraint. As testing for robust stability is computationally expensive, noticeably fewer ALNS iterations could be conducted. Thus, the solutions could not be improved sufficiently. Applying C3, less than half as many iterations (7,904) could be run on average than when applying C1 (20,521).

Table 5.33: Comparison of constraints sets by application of the hybrid algorithms ALNS×DBLF and ALNS×LS_DBLF; extended problem variants (3L-VRPTW/3L-VRPCB(TW), RL/SL, C1/C2/C3/C4/C5), (3L-VRPMB(TW)/3L-VRPSDP(TW), LSP/SL, C1/C2/C3/C4/C5); average TTDs, deviations from benchmark, deviations between TTDs, computing times, numbers of iterations; separated by packing heuristic and limited to instances with large items

		ALNS (large items)		
set	**KPI**	**DBLF**	**LS_DBLF**	**total**
C1	∅TTD*[DU]	1,601.31	1,531.37	1,562.27
	∅dev[%]	**0.00**	**0.00**	**0.00**
	max dev[%]	0.00	0.00	0.00
	∅devDBLF[%]	0.00	-9.60	
	∅ct[s]	189.44	1,453.38	894.99
	∅iter	20,688.66	20,388.61	20,521.17
C2	∅TTD[DU]	1,635.07	1,684.57	1,662.70
	∅dev[%]	2.13	9.73	6.37
	max dev[%]	16.03	44.43	44.43
	∅devDBLF[%]	0.00	-3.96	
	∅ct[s]	289.17	1,266.76	834.88
	∅iter	20,685.97	18,051.84	19,215.56
C3	∅TTD[DU]	1,644.80	1,702.39	1,676.95
	∅dev[%]	2.69	11.10	7.38
	max dev[%]	17.37	36.13	36.13
	∅devDBLF[%]	0.00	-2.80	
	∅ct[s]	2,799.22	3,079.93	2,955.92
	∅iter	10,637.68	5,740.13	7,903.79
C4	∅TTD[DU]	1,613.44	1,663.42	1,641.34
	∅dev[%]	1.02	8.60	5.25
	max dev[%]	63.46	66.41	66.41
	∅devDBLF[%]	0.00	-3.41	
	∅ct[s]	663.38	1,959.92	1,387.13
	∅iter	20,559.46	14,669.95	17,271.84
C5	∅TTD[DU]	1,709.24	1,755.95	1,735.31
	∅dev[%]	7.02	14.93	11.43
	max dev[%]	70.91	71.50	71.50
	∅devDBLF[%]	0.00	-2.96	
	∅ct[s]	3,075.31	3,230.99	3,162.21
	∅iter	7,904.60	4,265.96	5,873.45

∅: average, *: benchmark, ct: computing time per instance and run, dev: deviation from TTDs of benchmarks, devDBLF: deviation of TTDs from those of ALNS×DBLF, DU: distance units, iter: total number of iterations, KPI: key performance indicator, max: maximum, TTD: total travel distance.
Minimum average benchmark deviations per column are bold-faced.

Considerable differences can be observed for the applied packing heuristics. Whereas the TTDs provided by ALNS×DBLF with C2-C5 deviate on average 3.21 % from the benchmarks, the TTDs obtained by ALNS×LS_DBLF deviate on average 11.09 % from them.[47] In addition, the deviations between the TTDs provided by ALNS×-DBLF and ALNS×LS_DBLF decrease significantly if the extended constraints are taken into account. Considering the ALNS×DBLF solutions as benchmarks, they amount to ca. −9.6 % for C1, −4.0 % for C2, −2.8 % for C3, −3.4 % for C3, and −3.0 % for C5. Within LS_DBLF, the heuristic DBLF is called multiple times (if necessary), leading to a larger impact of the increasing computational effort on the total computing times. Due to the time limit, the number of iterations that can be conducted in total is reduced stronger in the case of the ALNS×LS_DBLF than in the case of the ALNS×DBLF. The most extreme example is the application of C5, which is associated with the largest computational efforts. Utilizing the DBLF heuristic within the hybrid algorithm, the number of iterations is approximately halved compared to C1. However, only about one quarter of the number of iterations conducted with C1 could be performed when applying ALNS×LS_DBLF.

In some cases, negative deviations from the benchmark TTDs are obtained. That is, some solutions provided under consideration of the respective constraint sets are better than the benchmark solutions. In particular, the results of C2-C4 appear to be very close, if not better than the benchmark results, if ALNS×DBLF and LSP are applied. The corresponding results for ALNS×DBLF and the different loading approaches with respect to the solution quality are presented in Table 5.34. Applying ALNS×LS_DBLF, the average deviations between the TTDs of C2-C5 from those of C1 are exclusively positive. Due to the restricted loading space, the influence of the new constraints seems to diminish compared to the loading approaches providing the complete loading space. For example, the LSP does not allow to stack a large number of items (keep in mind that the item heights range from 20 % to 50 % of the complete loading space height H). Therefore, the load bearing strength constraint is less likely to be violated. The constraints can be even less restrictive than the original ones as, e.g., a comparatively light item classified as non-fragile might be placeable above a fragile item.

In conclusion, reachability and load bearing strength constraints should be considered in a 3L-VRPBTW and included in the solution approach if they are of practical

[47] These averaged values are not provided in Table 5.33.

relevance. Among the considered additional constraints, they are associated with the lowest computational efforts. Hence, their integration can be valuable in order to ensure practically feasible solutions and does not cause extreme prolongations of the running times of the hybrid algorithm. In contrast, the integration of the robust stability constraint is – at least with the implementation presented in this thesis – computationally very expensive. Considerable deteriorations of the solution quality compared to solutions without consideration of such a constraint can occur because the ALNS cannot improve solutions sufficiently within a given time limit. However, the experiments on random routes have shown that the robust stability constraint is hardly ever violated if a (simple) vertical stability constraint is satisfied. Thus, the implementation of the robust stability constraint can most likely be neglected. If it is indispensable, raising the time limit should be considered.

Table 5.34: Comparison of constraints sets for different loading approaches by application of ALNS×DBLF; extended problem variants (3L-VRPTW/3L-VRPCB(TW), RL/SL, C1/C2/C3/C4/C5), (3L-VRPMB(TW)/3L-VRPSDP(TW), LSP/SL, C1/C2/C3/C4/C5); average TTDs, deviations from benchmark; restricted to instances with large items

		ALNS×DBLF (large items)		
set	**KPI**	**RL**	**LSP**	**SL**
C1	∅TTD*[DU]	1,642.84	1,862.52	1,487.01
	∅dev[%]	**0.00**	0.00	**0.00**
	min dev[%]	0.00	0.00	0.00
C2	∅TTD[DU]	1,698.56	1,858.13	1,522.62
	∅dev[%]	3.23	−0.21	2.38
	min dev[%]	−2.13	−9.46	−2.28
C3	∅TTD[DU]	1,682.50	1,897.25	1,535.61
	∅dev[%]	2.24	1.85	3.22
	min dev[%]	−2.24	−9.46	−1.89
C4	∅TTD[DU]	1,658.96	1,852.86	1,504.73
	∅dev[%]	1.25	**−0.50**	1.43
	min dev[%]	−5.82	−5.75	−6.61
C5	∅TTD[DU]	1,781.42	1,907.88	1,600.82
	∅dev[%]	8.44	2.45	7.88
	min dev[%]	−1.12	−10.44	−3.52

∅: average, *: benchmark, dev: deviation from TTDs of benchmarks, DU: distance units, LSP: loading space partition, KPI: key performance indicator, min: minimum, RL: rear loading, SL: side loading, TTD: total travel distance.
Minimum average deviations per column are bold-faced.

Chapter 6

Summary, conclusions and outlook

Vehicle routing problems with three-dimensional loading constraints are difficult combinatorial optimization problems combining two NP-hard subproblems – the routing and the packing problem. Yet, they are of high practical relevance if the transported goods are bulky and their spatial dimensions cannot be neglected when loading them. In addition, incorporating the pickup of goods into the planning of delivery tours may not only contribute to decreasing the travelled distances, but also to reducing the number of vehicles required for all transportations and the number of empty runs. Consequently, substantial cost savings can be realized. In this thesis, seven variants of VRPs with three-dimensional loading constraints, backhauls and time windows have been considered: the 3L-VRP with time windows, 3L-VRP with clustered backhauls (with and without time windows), 3L-VRP with mixed backhauls (with and without time windows) and the 3L-VRP with simultaneous delivery and pickup (with and without time windows).

The problem variants with mixed backhauls and simultaneous delivery and pickup require the simultaneous transport of linehaul and backhaul items. Therefore, they are particularly challenging and have very rarely been regarded in the literature before. In order to generate feasible packing plans under consideration of various constraints, such as the LIFO constraint, and to avoid any reloading during the route, two different loading approaches – loading space partition (LSP) and side loading – were proposed. Two alternative loading approaches were also regarded for the 3L-VRPTW and 3L-VRPCB(TW). The first one is the rear loading approach, which is also the standard loading approach in the literature. In addition, side loading was also considered for those problem variants.

Mathematical models were presented for the considered problems. Some packing constraints have been formulated mathematically for the first time. However, due to their complexity, solving them exactly is not possible within reasonable computing times. Therefore, a hybrid heuristic solution approach was presented consisting of an ALNS for solving the routing subproblem and various (alternatively employed) heuristics for packing the customer items.

In accordance with the literature, the packing constraints proposed by Gendreau

et al. (2006) were applied for the greater part of the numerical experiments. In order to deal with more realistic and practically relevant problems, further difficult packing constraints were also considered, namely robust stability, load bearing strength and reachability. They have been formulated as part of the mathematical optimization model and implemented into the packing heuristics.

Extensive numerical experiments were conducted in order to evaluate the variations of the hybrid algorithm. That is, the performance (solution quality, computing time) of the hybrid algorithms with different packing heuristics was compared for different problem variants. Furthermore, the influence of different loading approaches, problem and instance characteristics, and the impact and necessity of additional packing constraints was analysed.

In the following, the main insights gained for research (Chapter 6.1) and for practice (Chapter 6.2) are summarized. This thesis concludes with an outlook to further research (Chapter 6.3).

6.1 Contributions to research

This thesis provides a comprehensive overview of 3L-VRPs with backhauls and time windows and one of the first systematic presentations of such problems. Seven different problem variants and – taking various loading approaches and sets of packing constraints into account – 70 extended problem variants are regarded.

A consistent and detailed problem description was provided for various problem variants. This includes the formulation of routing and packing constraints as well as constraints concerning entire solutions. Moreover, the concept of a packing plan with a different packing pattern at every stage of a route was introduced. Ten different packing constraints were considered including the constraint set proposed by Gendreau et al. (2006) and reachability, robust stability and load bearing strength constraints.

Mathematical optimization models were presented for the regarded problem variants. They are based on 3L-CVRP and CLP models previously proposed in the literature (in particular by Junqueira et al., 2012a,b, 2013). Those models were adapted to the studied problem variants. Some packing constraints were reformulated as they were defined differently in this thesis. New formulations were proposed for some constraints. For example, load bearing strength constraints previously regarded

in the literature usually follow the assumptions proposed by Ratcliff and Bischoff (1998) that the weight of an item is passed straight downwards. In this thesis, the constraint was modelled and implemented in a way that the actual physical settings are taken into account. That is, each item transmits its own weight and the weights of items placed above it evenly over its bottom face. Hence, the proposed model can better depict real-world problems. The formulation of this variant of the load bearing strength in the mathematical model as well as its implementation into the packing heuristics is non-trivial. Mathematical formulations for the reachability and robust stability constraint were – to the best of the author's knowledge – proposed for the first time.

A unified hybrid solution approach was presented for solving the variants of the 3L-VRPBTW. An ALNS heuristic was used for solving the routing subproblem. This approach is based on the works of Ropke and Pisinger (2006a,b) and was extended by new operators and further modifications. In order to determine the most suitable packing heuristic for the hybrid approach, numerous packing construction heuristics (different variants of deepest-bottom-left-fill (DBLF) and touching area heuristics) were tested. Among the tested heuristics, the DBLF heuristic proved superior. Three different packing heuristics were subsequently integrated into the ALNS for the hybrid algorithm: the DBLF heuristic, a local search-based procedure with an open space packing heuristic (LS_OS; cf. Zhang et al., 2015) and the DBLF heuristic embedded in the LS framework (LS_DBLF).

One part of the numerical experiments consisted in applying the ALNS to well-known (one-dimensional) VRPBTW benchmark instance sets from the literature in order to evaluate its performance. In the course of this, very good results were obtained and in multiple cases the best known solutions of the benchmark instances could even be improved. Thus, it can be assumed that the proposed extensions and modifications constitute valuable additions to the original approach. The modifications include a larger number of removal heuristics, construction of the initial solution by means of the savings heuristic, an adapted Shaw removal heuristic and a modified weight adjustment procedure. In addition, new removal heuristics were proposed. Experiments suggest that the use of multiple removal heuristics contributes to a better performance of the ALNS. Choosing a well balanced set of approximately ten heuristics that are able to react to a variety of instance characteristics is recommended. A larger number of heuristics does not contribute significantly to the

solution quality.

In order to examine the resulting variants of the hybrid algorithm, new benchmark instances were provided for various 3L-VRPBTWs. They cover a wide range of instance characteristics, like different item sizes, numbers of customers or time window widths.

In the experiments with the hybrid algorithms, the approaches including the LS-based packing procedures outperformed the hybrid algorithm with the DBLF heuristic. Over all problem classes, the results provided by ALNS×LS_DBLF and ALNS×LS_OS do not differ much. ALNS×LS_DBLF usually obtained the best results when a rear loading approach was used, i.e. either the standard rear loading, the loading space partition or the side loading implemented as a rear loading approach by swapping the loading space length and width. In contrast, ALNS×LS_OS performed (on average) best when the side loading approach for the simultaneous transport of linehaul and backhaul items was applied. This loading approach includes a modified LIFO constraint. For example, a linehaul item must not be placed in front of, above or to the right of another linehaul that is delivered earlier. Applying a DBLF heuristic to a packing problem with such a LIFO constraint, the resulting packing patterns often contain large gaps and are, thus, inefficient. The results could be improved by modifying the DBLF heuristics although the solution quality of LS_OS was not reached. Within the open space heuristic, items are placed according to a deepest-left-bottom priority, which appears to be more suitable for the regarded side loading approach.

The computational efforts of the hybrid algorithms can be crucial for the solution quality. The differences between ALNS×DBLF and the approaches with the LS-based packing heuristics are largest if the instance and problem characteristics induce comparatively low computing time requirements. Such instance characteristics are, e.g., large items or narrow time windows. In these cases, enough ALNS iterations can be conducted within a given time limit in order to improve the initial solutions sufficiently. Hence, the differences in the solution quality are caused by the performances of the integrated packing heuristics and the benefits of using more sophisticated approaches – like an LS-based approach – can be exploited. In contrast, solving other instances, e.g. with small items, proved to be computationally expensive. Subsequently, procedures requiring more time for solving one packing problem can conduct considerably fewer iterations in total and result in solutions

that are in some cases worse than those obtained by ALNS×DBLF.
Comparing the computing times for the hybrid algorithms for 3L-VRPBTWs and the ALNS for one-dimensional VRPBTWs, it becomes evident that the execution of the packing heuristics (which are called several tens of thousands of times during the search) require up to 99 % of the total computing time of the hybrid algorithm. Thus, the implementation of very fast packing heuristics and an efficient integration of them into the routing algorithm is vital for obtaining good solutions. It should be aimed at calling the packing procedure as rarely as possible. Implementing a cache, which stores routes that have already been packed, can be one mean for reducing the packing effort.

6.2 Managerial insights

Based on the experiments, several conclusions can be drawn for practice. They concern the application of a solution approach for the integrated routing and packing problem, the benefits of considering backhauls in the planning of delivery tours, recommendations of loading approaches and the consideration of further practically relevant packing constraints. In the following, these aspects are discussed.
A hybrid solution approach was presented that can be applied for a broad range of 3L-VRPs with backhauls and time windows and can solve the integrated routing and packing problem within reasonable computing times. Thus, it is realistic that such an approach can be implemented for the operative planning of routes. In contrast to solution approaches for (one-dimensional) VRPs, the hybrid algorithm does not only provide routes but also a packing plan for each route. The experiments have shown the necessity of regarding the routing and packing problem in an integrated way. In particular, it is of great significance if the items are large. The solutions provided by applying the approach to the 3L-VRPBTW instances differ considerably from solutions obtained by reducing the instances to one-dimensional problems, i.e. taking only weight and volume capacity constraints into account and disregarding loading constraints. Thus, it can be assumed that the 1D solutions are most likely not feasible if the actual item dimensions have to be considered for the loading. As opposed to this, no greater deviations could be observed in the case of small items. It can be assumed that the routes obtained by solving a one-dimensional VRPBTW can often be executed also under consideration of 3D loading constraints as small

items are easier to pack. In addition, solving a 3L-VRPBTW with small items is computationally very expensive. Based on this, solving an integrated routing and packing problem might not always be justified if the items are small. However, the benefits of providing solutions quickly must be weighted against automatically obtaining feasible packing plans. If the items are large, the packing problems are very difficult and the integrated planning of routing and packing is inevitable in order to guarantee feasible solutions.

Furthermore, the benefits of making use of backhaul transportations were analysed. For this purpose, the solutions of 3L-VRPCB(TW) and 3L-VRPMB(TW) instances were compared. An additional problem variant was regarded exclusively for these experiments: linehaul and backhaul customers must be visited in separated tours. A strict separation of deliveries and pickups can cause significant increases in the total travel distances in comparison to incorporating backhauls into the delivery tours. Here, they amounted to about 17 % compared to the TTDs of 3L-VRPCB(TW) solutions. Moreover, allowing mixed visiting sequences of linehaul and backhaul customers (3L-VRPMB(TW)) is usually more beneficial than clustering them in a route (3L-VRPCB(TW)). This is particularly true if the transported items are small. If they are large, however, the benefits diminish considerably as large items are more difficult to arrange in a way that linehaul and backhaul items do not obstruct each others loading and unloading. Hence, in this case it can be recommended to impose the restriction to visit all linehaul customers before the first backhaul customer in a route in order to facilitate (un)loading and avoid any reloading.

Different loading approaches were implemented and tested for the different problem variants. If either no backhaul items are present, or linehaul and backhaul items are transported completely separately (3L-VRPTW, 3L-VRPCB(TW)), no significant differences can be observed between the rear and side loading approaches. In contrast, the side loading approach is very beneficial (in comparison to the LSP approach) if linehaul and backhaul items are transported simultaneously. This could be expected as the approach allows for utilizing the whole loading space. The benefits of applying the side loading are particularly large if the transported items are large and if the ratio of linehaul and backhaul items is uneven.

The impact and necessity of additional packing constraints, which might be relevant in practice, was evaluated. These experiments have shown that integrating the load bearing strength and reachability constraints is sensible if they are required. The

solutions differ significantly from those obtained without their consideration and additional computational efforts are negligible. Therefore, if those constraints need to be taken into account, it appears to be inevitable to implement them. In contrast, checking for robust stability leads to considerable increases in computing times although the vast majority of the solutions provided without explicitly considering the constraint do satisfy it. Hence, implementing the robust stability constraint is not recommended especially considering that possible smaller changes in the packing plan can most likely be made manually.

6.3 Outlook for future research

The examined packing constraints constitute only a part of the restrictions to be considered in practice. In order to increase the planning accuracy, i.e. to guarantee the feasibility of the solutions with regard to all practically relevant constraints, the integration and implementation of further constraints could be interesting for future research, too. This can include the balancing of the weight of the load within the loading space, the horizontal (dynamic) stability or the consideration of the axle load distribution, which is usually subject to road traffic laws.

Furthermore, better solutions, higher utilization rates and more flexibility might be obtained if reloading was allowed during the routes. Here, the LIFO constraints are very strict and have a huge influence on the packing plans. Two aspects must be considered when reloading is allowed: (i) the time required for unloading items that block other items must be accounted for (for example, relative to the weight or volume of the moved items), and (ii) the reloaded item must be assigned a new position in the loading space.

Moreover, further loading approaches for realizing the simultaneous transport of linehaul and backhaul items can be examined. An extension of the loading space partition approach presented here could consist in considering a flexible separation of the loading space. Provided that the loading spaces are separated into sections of equal size, the results of the numerical experiments lead to the assumption that the LSP approach is particularly impractical if the ratio of linehaul and backhaul items is uneven. In this case, the possibility of adjusting the separation to the actual ratio could make this approach more beneficial. Furthermore, it might be interesting to consider this kind of problems with a heterogeneous vehicle fleet. As

the experiments have shown, a large proportion of the routes of 3L-VRPMB(TW) solutions contain only one type of customers (linehaul or backhaul) if the numbers of linehaul and backhaul customers differ considerably. Using a heterogeneous fleet, such a problem could be solved by using a double decker vehicle for routes with a mixture of linehaul and backhaul customers, and a vehicle without LSP (rear or side loaded) for routes with only one kind of customers.

The problems considered in this thesis are static and deterministic. Deviating from that, problem variants with dynamic demands may be considered. That is, further requests turn up during the execution of a route, e.g. additional goods need to be picked up. This situation would not only require an adaptation of the route, but also an incorporation of additional items into the packing plan. An important prerequisite is an efficient and effective solution approach that generates good, feasible solutions within very short computing times. Stochasticity can be regarded, for example, concerning the item sizes. In particular, if items are to be picked up, the information about the item sizes can be incorrect. The edges might be measured falsely or merely estimated. Thus, designated solutions need to be sufficiently robust in order to cope with such inaccuracies. Furthermore, the execution of a route can be influenced by stochastic processes. In particular, the required driving times cannot always be predicted reliably. In this context, a robust planning aims at minimizing the probability of violating time windows.

Moreover, rather simple packing heuristics were employed in this thesis because strong computational requirements prevent the use of more sophisticated CLP approaches. Nevertheless, technological advances and more elaborated integrations of routing and packing algorithms may enable using such approaches in the future.

Bibliography

Allen, S. D.; Burke, E. K.; Kendall, G. (2011): A hybrid placement strategy for the three-dimensional strip packing problem. In: *European Journal of Operational Research*, 209, 3, pp. 219–227.

Altınel, İ. K.; Öncan, T. (2005): A new enhancement of the Clarke and Wright savings heuristic for the capacitated vehicle routing problem. In: *Journal of the Operational Research Society*, 56, 8, pp. 954–961.

Angelelli, E.; Mansini, R. (2002): The Vehicle Routing Problem with Time Windows and Simultaneous Pick-up and Delivery. In: A. Klose; M. G. Speranza; L. N. van Wassenhove, eds., *Quantitative Approaches to Distribution Logistics and Supply Chain Management*, pp. 249–267, Springer Berlin Heidelberg, Berlin, Heidelberg.

Anily, S. (1996): The vehicle-routing problem with delivery and back-haul options. In: *Naval Research Logistics*, 43, 3, pp. 415–434.

Aprile, D.; Egeblad, J.; Garavelli, A. C.; Lisi, S.; Pisinger, D. (2007): Logistics optimization: vehicle routing with loading constraints. In: *ICPR-19th International Conference on Production Research*.

Araya, I.; Riff, M.-C. (2014): A beam search approach to the container loading problem. In: *Computers & Operations Research*, 43, pp. 100–107.

Attanasio, A.; Fuduli, A.; Ghiani, G.; Triki, C. (2007): Integrated Shipment Dispatching and Packing Problems: A Case Study. In: *Journal of Mathematical Modelling and Algorithms*, 6, 1, pp. 77–85.

Avci, M.; Topaloglu, S. (2015): An adaptive local search algorithm for vehicle routing problem with simultaneous and mixed pickups and deliveries. In: *Computers & Industrial Engineering*, 83, pp. 15–29.

Avci, M.; Topaloglu, S. (2016): A hybrid metaheuristic algorithm for heterogeneous vehicle routing problem with simultaneous pickup and delivery. In: *Expert Systems with Applications*, 53, pp. 160–171.

Baker, B. S.; Coffman, Jr., E. G.; Rivest, R. L. (1980): Orthogonal Packings in Two Dimensions. In: *SIAM Journal on Computing*, 9, 4, pp. 846–855.

Baldacci, R.; Mingozzi, A.; Roberti, R. (2011): New Route Relaxation and Pricing Strategies for the Vehicle Routing Problem. In: *Operations Research*, 59, 5, pp. 1269–1283.

Bartók, T.; Imreh, C. (2011): Pickup and Delivery Vehicle Routing with Multidimensional Loading Constraints. In: *Acta Cybernetica*, 20, 1, pp. 17–33.

Belloso, J.; Juan, A. A.; Faulin, J.; Serrano, A. (2015): Using multi-start biased randomization of heuristics to solve the vehicle routing problem with clustered backhauls. In: K. Sheibani; P. Hirsch; T. E. Nordlander; R. Montemanni; S. Sofianopoulou; J. Faulin, eds., *Lecture Notes in Management Science*, p. 16, Vienna.

Belmecheri, F.; Prins, C.; Yalaoui, F.; Amodeo, L. (2013): Particle swarm optimization algorithm for a vehicle routing problem with heterogeneous fleet, mixed backhauls, and time windows. In: *Journal of Intelligent Manufacturing*, 24, 4, pp. 775–789.

Bent, R.; van Hentenryck, P. (2004): A Two-Stage Hybrid Local Search for the Vehicle Routing Problem with Time Windows. In: *Transportation Science*, 38, 4, pp. 515–530.

Bianchessi, N.; Righini, G. (2007): Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery. In: *Computers & Operations Research*, 34, 2, pp. 578–594.

Bischoff, E. E. (2006): Three-dimensional packing of items with limited load bearing strength. In: *European Journal of Operational Research*, 168, 3, pp. 952–966.

Bischoff, E. E.; Janetz, F.; Ratcliff, M. (1995): Loading pallets with non-identical items. In: *European Journal of Operational Research*, 84, 3, pp. 681–692.

Bischoff, E. E.; Ratcliff, M. (1995): Issues in the development of approaches to container loading. In: *Omega*, 23, 4, pp. 377–390.

Böge, A.; Böge, W., eds. (2017): *Handbuch Maschinenbau: Grundlagen und Anwendungen der Maschinenbau-Technik*. Springer Vieweg, Wiesbaden, 23., überarbeitete auflage edn.

Bortfeldt, A. (2012): A hybrid algorithm for the capacitated vehicle routing problem with three-dimensional loading constraints. In: *Computers & Operations Research*, 39, 9, pp. 2248–2257.

Bortfeldt, A.; Gehring, H. (2001): A hybrid genetic algorithm for the container loading problem. In: *European Journal of Operational Research*, 131, 1, pp. 143–161.

Bortfeldt, A.; Gehring, H.; Mack, D. (2003): A parallel tabu search algorithm for solving the container loading problem. In: *Parallel Computing*, 29, 5, pp. 641–662.

Bortfeldt, A.; Hahn, T.; Männel, D.; Mönch, L. (2015): Hybrid algorithms for the vehicle routing problem with clustered backhauls and 3D loading constraints. In: *European Journal of Operational Research*, 243, 1, pp. 82–96.

Bortfeldt, A.; Homberger, J. (2013): Packing first, routing second—a heuristic for the vehicle routing and loading problem. In: *Computers & Operations Research*, 40, 3, pp. 873–885.

Bortfeldt, A.; Wäscher, G. (2013): Constraints in container loading – A state-of-the-art review. In: *European Journal of Operational Research*, 229, 1, pp. 1–20.

Brandão, J. (2006): A new tabu search algorithm for the vehicle routing problem with backhauls. In: *European Journal of Operational Research*, 173, 2, pp. 540–555.

Casco, D. O.; Golden, B. L.; Wasil, E. A. (1988): Vehicle routing with backhauls: Models, algorithms, and case studies. In: B. L. Golden, ed., *Vehicle routing*, Studies in management science and systems, pp. 127–147, North-Holland, Amsterdam u.a.

Çatay, B. (2010): A new saving-based ant algorithm for the Vehicle Routing Problem with Simultaneous Pickup and Delivery. In: *Expert Systems with Applications*, 37, 10, pp. 6809–6817.

Ceschia, S.; Schaerf, A.; Stützle, T. (2013): Local search techniques for a routing-packing problem. In: *Computers & Industrial Engineering*, 66, 4, pp. 1138–1149.

Chabrier, A. (2006): Vehicle Routing Problem with elementary shortest path based column generation. In: *Computers & Operations Research*, 33, 10, pp. 2972–2990.

Chen, C. S.; Lee, S. M.; Shen, Q. S. (1995): An analytical model for the container loading problem. In: *European Journal of Operational Research*, 80, 1, pp. 68–76.

Chen, J. F.; Wu, T. H. (2006): Vehicle routing problem with simultaneous deliveries and pickups. In: *Journal of the Operational Research Society*, 57, 5, pp. 579–587.

Christensen, S. G.; Rousøe, D. M. (2009): Container loading with multi-drop constraints. In: *International Transactions in Operational Research*, 16, 6, pp. 727–743.

Christofides, N.; Eilon, S. (1969): An Algorithm for the Vehicle-dispatching Problem. In: *Journal of the Operational Research Society*, 20, 3, pp. 309–318.

Christofides, N.; Mingozzi, A.; Toth, P. (1979): The vehicle routing problem. In: N. Christofides; A. Mingozzi; P. Toth; C. Sandi, eds., *Combinatorial optimization*, A Wiley-Interscience publiction, Wiley, Chichester.

Christofides, N.; Mingozzi, A.; Toth, P. (1981): State-space relaxation procedures for the computation of bounds to routing problems. In: *Networks*, 11, 2, pp. 145–164.

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

Cook, W.; Rich, J. L. (1999): A parallel cutting-plane algorithm for the vehicle routing problem with time windows. In: *Computational and Applied Mathematics Department, Rice University, Houston, TX, Technical Report*.

Côté, J. F.; Guastaroba, G.; Speranza, M. G. (2017): The value of integrating loading and routing. In: *European Journal of Operational Research*, 257, 1, pp. 89–105.

Crispim, J.; Brandão, J. (2001): Reactive tabu search and variable neighbourhood descent applied to the vehicle routing problem with backhauls. In: *MIC'2001–4th Metaheuristics International Conference*, pp. 631–636.

Crispim, J.; Brandão, J. (2005): Metaheuristics Applied to Mixed and Simultaneous Extensions of Vehicle Routing Problems with Backhauls. In: *The Journal of the Operational Research Society*, 56, 11, pp. 1296–1302.

Dantzig, G. B.; Ramser, J. H. (1959): The Truck Dispatching Problem. In: *Management Science*, 6, 1, pp. 80–91.

Davies, A.; Bischoff, E. E. (1999): Weight distribution considerations in container loading. In: *European Journal of Operational Research*, 114, 3, pp. 509–527.

Deif, I.; Bodin, L. (1984): Extension of the Clarke and Wright algorithm for solving the vehicle routing problem with backhauling. In: *Proceedings of the Babson conference on software uses in transportation and logistics management*, pp. 75–96.

Dell'Amico, M.; Righini, G.; Salani, M. (2006): A Branch-and-Price Approach to the Vehicle Routing Problem with Simultaneous Distribution and Collection. In: *Transportation Science*, 40, 2, pp. 235–247.

Demir, E.; Bektaş, T.; Laporte, G. (2012): An adaptive large neighborhood search heuristic for the Pollution-Routing Problem. In: *European Journal of Operational Research*, 223, 2, pp. 346–359.

Derigs, U.; Metz, A. (1992): A matching-based approach for solving a delivery/pick-up vehicle routing problem with time constraints. In: *OR Spektrum*, 14, 2, pp. 91–106.

Desaulniers, G.; Lessard, F.; Hadjar, A. (2008): Tabu Search, Partial Elementarity, and Generalized k -Path Inequalities for the Vehicle Routing Problem with Time Windows. In: *Transportation Science*, 42, 3, pp. 387–404.

Dethloff, J. (2001): Vehicle routing and reverse logistics: The vehicle routing problem with simultaneous delivery and pick-up. In: *OR Spektrum*, 23, 1, pp. 79–96.

Dethloff, J. (2002): Relation between Vehicle Routing Problems: An Insertion Heuristic for the Vehicle Routing Problem with Simultaneous Delivery and Pick-Up Applied to the Vehicle Routing Problem with Backhauls. In: *The Journal of the Operational Research Society*, 53, 1, pp. 115–118.

Diestel, R. (2000): *Graph theory*, vol. 173 of *Graduate texts in mathematics*. Springer, New York, NY, 2. ed. edn.

Dominguez, O.; Guimarans, D.; Juan, A. A. (2015): A Hybrid Heuristic for the 2L-VRP with Clustered Backhauls. In: *Proceedings of the XVI Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA)*.

Domschke, W.; Drexl, A.; Klein, R.; Scholl, A. (2015): *Einführung in Operations Research*. Springer Berlin Heidelberg, Berlin, Heidelberg.

Duhamel, C.; Potvin, J.-Y.; Rousseau, J.-M. (1997): A Tabu Search Heuristic for the Vehicle Routing Problem with Backhauls and Time Windows. In: *Transportation Science*, 31, 1, pp. 49–59.

Eley, M. (2002): Solving container loading problems by block arrangement. In: *European Journal of Operational Research*, 141, 2, pp. 393–409.

Escobar-Falcón, L. M.; Álvarez-Martínez, D.; Granada-Echeverri, M.; Escobar, J. W.; Romero-Lázaro, R. A. (2016): A matheuristic algorithm for the three-dimensional loading capacitated vehicle routing problem (3L-CVRP). In: *Revista Facultad de Ingeniería Universidad de Antioquia*, 78, pp. 9–20.

eurostat (2017): Summary of annual road freight transport by type of operation and type of transport (1 000 t, Mio Tkm, Mio Veh-km), URL: `http://appsso.eurostat.ec.europa.eu/nui/show.do?dataset=road_go_ta_tott&lang=en` (Accessed: 13/11/2017).

Fan, J. (2011): The Vehicle Routing Problem with Simultaneous Pickup and Delivery Based on Customer Satisfaction. In: *Procedia Engineering*, 15, pp. 5284–5289.

Fanslau, T.; Bortfeldt, A. (2010): A Tree Search Algorithm for Solving the Container Loading Problem. In: *INFORMS Journal on Computing*, 22, 2, pp. 222–235.

Fasano, G. (1999): Cargo Analytical Integration in Space Engineering: A Three-dimensional Packing Model. In: T. A. Ciriani; S. Gliozzi; E. L. Johnson; R. Tadei, eds., *Operational Research in Industry*, pp. 232–246, Palgrave Macmillan UK, London.

Fekete, S. P.; Schepers, J. (1997): A new exact algorithm for general orthogonal d-dimensional knapsack problems. In: R. Burkard; G. Woeginger, eds., *Algorithms - ESA '97*, vol. 1284 of *Lecture Notes in Computer Science*, pp. 144–156, Springer, Berlin and Heidelberg.

Fekete, S. P.; Schepers, J. (2004): A Combinatorial Characterization of Higher-Dimensional Orthogonal Packing. In: *Mathematics of Operations Research*, 29, 2, pp. 353–368.

Fekete, S. P.; Schepers, J.; van der Veen, J. C. (2007): An Exact Algorithm for Higher-Dimensional Orthogonal Packing. In: *Operations Research*, 55, 3, pp. 569–587.

Fox, K. R.; Gavish, B.; Graves, S. C. (1980): Technical Note—An n-Constraint Formulation of the (Time-Dependent) Traveling Salesman Problem. In: *Operations Research*, 28, 4, pp. 1018–1021.

Fuellerer, G.; Doerner, K. F.; Hartl, R. F.; Iori, M. (2010): Metaheuristics for vehicle routing problems with three-dimensional loading constraints. In: *European Journal of Operational Research*, 201, 3, pp. 751–759.

Gajpal, Y.; Abad, P. L. (2009): Multi-ant colony system (MACS) for a vehicle routing problem with backhauls. In: *European Journal of Operational Research*, 196, 1, pp. 102–117.

Ganesh, K.; Narendran, T. T. (2007): CLOVES: A cluster-and-search heuristic to solve the vehicle routing problem with delivery and pick-up. In: *European Journal of Operational Research*, 178, 3, pp. 699–717.

Garcia, B.-L.; Potvin, J.-Y.; Rousseau, J.-M. (1994): A parallel implementation of the Tabu search heuristic for vehicle routing problems with time window constraints. In: *Computers & Operations Research*, 21, 9, pp. 1025–1033.

García-Nájera, A.; Bullinaria, J. A.; Gutiérrez-Andrade, M. A. (2015): An evolutionary approach for multi-objective vehicle routing problems with backhauls. In: *Computers & Industrial Engineering*, 81, pp. 90–108.

Gehring, H.; Bortfeldt, A. (1997): A Genetic Algorithm for Solving the Container Loading Problem. In: *International Transactions in Operational Research*, 4, 5-6, pp. 401–418.

Gehring, H.; Homberger, J. (1999): A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In: *Proceedings of EUROGEN99*, vol. 2, pp. 57–64, Springer, Berlin.

Gehring, H.; Menschner, K.; Meyer, M. (1990): A computer-based heuristic for packing pooled shipment containers. In: *European Journal of Operational Research*, 44, 2, pp. 277–288.

Gélinas, S.; Desrochers, M.; Desrosiers, J.; Solomon, M. M. (1995): A new branching strategy for time constrained routing problems with application to backhauling. In: *Annals of Operations Research*, 61, 1, pp. 91–109.

Gendreau, M.; Iori, M.; Laporte, G.; Martello, S. (2006): A Tabu Search Algorithm for a Routing and Container Loading Problem. In: *Transportation Science*, 40, 3, pp. 342–350.

George, J. A.; Robinson, D. F. (1980): A heuristic for packing boxes into a container. In: *Computers & Operations Research*, 7, 3, pp. 147–156.

Ghaziri, H.; Osman, I. H. (2006): Self-organizing feature maps for the vehicle routing problem with backhauls. In: *Journal of Scheduling*, 9, 2, pp. 97–114.

Gilmore, P. C.; Gomory, R. E. (1965): Multistage Cutting Stock Problems of Two and More Dimensions. In: *Operations Research*, 13, 1, pp. 94–120.

Goetschalckx, M.; Jacobs-Blecha, C. (1989): The vehicle routing problem with backhauls. In: *European Journal of Operational Research*, 42, 1, pp. 39–51.

Goetschalckx, M.; Jacobs-Blecha, C. (1993): The vehicle routing problem with backhauls: Properties and solution algorithms. In: *Technical Report MHRC-TR-88-13, Georgia Institute*.

Goksal, F. P.; Karaoglan, I.; Altiparmak, F. (2013): A hybrid discrete particle swarm optimization for vehicle routing problem with simultaneous pickup and delivery. In: *Computers & Industrial Engineering*, 65, 1, pp. 39–53.

Golden, B.; Assad, A.; Levy, L.; Gheysens, F. (1984): The fleet size and mix vehicle routing problem. In: *Computers & Operations Research*, 11, 1, pp. 49–66.

Golden, B. L.; Baker, E.; Alfaro, J.; Schaffer, J. (1985): The vehicle routing problem with backhauling: two approaches. In: Hammesfahr; RD, eds., *Proceedings of the twenty-first annual meeting of the S.E. TIMS, Myrtle Beach, SC, USA*.

Gonçalves, J. F.; Resende, M. G. (2012): A parallel multi-population biased random-key genetic algorithm for a container loading problem. In: *Computers & Operations Research*, 39, 2, pp. 179–190.

Halse, K. (1992): *Modeling and solving complex vehicle routing problems. Ph.D. Thesis*, Technical University of Denmark, Lyngby.

Hasama, T.; Kokubugata, H.; Kawashima, H. (1998): A heuristic approach based on the string model to solve vehicle routing problem with backhauls. In: *Proceedings of the 5th World Congress on Intelligent Transport Systems*.

Hemminki, J. (1994): *Container loading with variable strategies in each layer*. University of Turku.

Hezer, S.; Kara, Y. (2011): Solving vehicle routing problem with simultaneous delivery and pick-up using bacterial foraging optimization algorithm. In: *Proceeding of the 41st International Conference on Computers & Industrial Engineering 2011*, Curran, Red Hook, NY.

Hifi, M. (2002): Approximate algorithms for the container loading problem. In: *International Transactions in Operational Research*, 9, 6, pp. 747–774.

Homberger, J. (2000): *Verteilt-parallele Metaheuristiken zur Tourenplanung: Lösungsverfahren für das Standardproblem mit Zeitfensterrestriktionen: Zugl.: Hagen, Fernuniv., Diss., 2000*. Gabler Edition Wissenschaft, Dt. Univ.-Verl., Wiesbaden, 1. aufl. edn.

Homberger, J.; Gehring, H. (1999): Two Evolutionary Metaheuristics For The Vehicle Routing Problem With Time Windows. In: *INFOR: Information Systems and Operational Research*, 37, 3, pp. 297–318.

Hopper, E. (2000): *Two-dimensional packing utilising evolutionary algorithms and other meta-heuristic methods*. Ph.D. thesis, University of Wales. Cardiff.

Iori, M.; Martello, S. (2010): Routing problems with loading constraints. In: *TOP*, 18, 1, pp. 4–27.

Irnich, S.; Schneider, M.; Vigo, D. (2014a): Four Variants of the Vehicle Routing Problem. In: P. Toth; D. Vigo, eds., *Vehicle routing*, MOS-SIAM series on optimization, SIAM, Philadelphia, PA.

Irnich, S.; Toth, P.; Vigo, D. (2014b): The Family of Vehicle Routing Problems. In: P. Toth; D. Vigo, eds., *Vehicle routing*, MOS-SIAM series on optimization, pp. 1–33, SIAM, Philadelphia, PA.

Irnich, S.; Villeneuve, D. (2006): The Shortest-Path Problem with Resource Constraints and k -Cycle Elimination for k $\geq$ 3. In: *INFORMS Journal on Computing*, 18, 3, pp. 391–406.

Jakobs, S. (1996): On genetic algorithms for the packing of polygons. In: *European Journal of Operational Research*, 88, 1, pp. 165–181.

Jepsen, M.; Petersen, B.; Spoorendonk, S.; Pisinger, D. (2008): Subset-Row Inequalities Applied to the Vehicle-Routing Problem with Time Windows. In: *Operations Research*, 56, 2, pp. 497–511.

Jun, Y.; Kim, B.-I. (2012): New best solutions to VRPSPD benchmark problems by a perturbation based algorithm. In: *Expert Systems with Applications*, 39, 5, pp. 5641–5648.

Junqueira, L.; Morabito, R.; Sato Yamashita, D. (2012a): MIP-based approaches for the container loading problem with multi-drop constraints. In: *Annals of Operations Research*, 199, 1, pp. 51–75.

Junqueira, L.; Morabito, R.; Sato Yamashita, D. (2012b): Three-dimensional container loading models with cargo stability and load bearing constraints. In: *Computers & Operations Research*, 39, 1, pp. 74–85.

Junqueira, L.; Oliveira, J. F.; Carravilla, M. A.; Morabito, R. (2013): An optimization model for the vehicle routing problem with practical three-dimensional loading constraints. In: *International Transactions in Operational Research*, 20, 5, pp. 645–666.

Kallehauge, B.; Larsen, J.; Madsen, O. B. (2000): Lagrangean duality and non-differentiable optimization applied on routing with time windows-experimental results. In: *Relatório interno IMM-REP-2000-8, Department of Mathematical Modeling, Technical University of Denmark, Lyngby, Dinamarca*.

Karabulut, K.; İnceoğlu, M. M. (2005): A Hybrid Genetic Algorithm for Packing in 3D with Deepest Bottom Left with Fill Method. In: T. Yakhno, ed., *Advances in Information Systems*, vol. 3261 of *Lecture Notes in Computer Science*, pp. 441–450, Springer-Verlag Berlin/Heidelberg, Berlin, Heidelberg.

Klug, F. (2010): *Logistikmanagement in der Automobilindustrie: Grundlagen der Logistik im Automobilbau*. VDI-Buch, Springer, Berlin.

Knight, K. W.; Hofer, J. P. (1968): Vehicle Scheduling with Timed and Connected Calls: A Case Study. In: *Journal of the Operational Research Society*, 19, S3, pp. 299–310.

Kohl, N.; Desrosiers, J.; Madsen, O. B. G.; Solomon, M. M.; Soumis, F. (1999): 2-Path Cuts for the Vehicle Routing Problem with Time Windows. In: *Transportation Science*, 33, 1, pp. 101–116.

Kolen, A. W. J.; Rinnooy Kan, A. H. G.; Trienekens, H. W. J. M. (1987): Vehicle Routing with Time Windows. In: *Operations Research*, 35, 2, pp. 266–273.

Kontoravdis, G.; Bard, J. F. (1995): A GRASP for the Vehicle Routing Problem with Time Windows. In: *ORSA Journal on Computing*, 7, 1, pp. 10–23.

Kopp, D. (2015): *Ein Packverfahren für das Capacitated-Vehicle-Routing-Problem mit 3D-Loading-Constraints auf der Basis des Bottom-Left-Fill-Algorithmus mit Implementierung. Master thesis*, Otto-von-Guericke University, Magdeburg.

Krebs, C. (2017): *Das Capacitated-Vehicle-Routing-Problem mit komplexen 3D-Laderestriktionen. Master thesis*, Otto-von-Guericke University, Magdeburg.

Kruskal, J. B. (1956): On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. In: *Proceedings of the American Mathematical Society*, 7, 1, pp. 48.

Lacomme, P.; Toussaint, H.; Duhamel, C. (2013): A GRASP×ELS for the vehicle routing problem with basic three-dimensional loading constraints. In: *Engineering Applications of Artificial Intelligence*, 26, 8, pp. 1795–1810.

Larsen, J. (1999): *Parallelization of the vehicle routing problem with time windows*. Ph.D. thesis, Technical University of Denmark, Department of Informatics and Mathematical Modeling.

Lenstra, J. K.; Kan, A. H. G. R. (1981): Complexity of vehicle routing and scheduling problems. In: *Networks*, 11, 2, pp. 221–227.

Li, J.; Pardalos, P. M.; Sun, H.; Pei, J.; Zhang, Y. (2015): Iterated local search embedded adaptive neighborhood selection approach for the multi-depot vehicle routing problem with simultaneous deliveries and pickups. In: *Expert Systems with Applications*, 42, 7, pp. 3551–3561.

Liu, D.; Teng, H. (1999): An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles. In: *European Journal of Operational Research*, 112, 2, pp. 413–420.

Lodi, A.; Martello, S.; Vigo, D. (1999): Heuristic and Metaheuristic Approaches for a Class of Two-Dimensional Bin Packing Problems. In: *INFORMS Journal on Computing*, 11, 4, pp. 345–357.

Ma, H.-w.; Zhu, W.; Xu, S. (2011): Research on the Algorithm for 3L-CVRP with Considering the Utilization Rate of Vehicles. In: R. Chen, ed., *Intelligent Computing and Information Science*, vol. 134 of *Communications in Computer and Information Science*, pp. 621–629, Springer Berlin Heidelberg, Berlin, Heidelberg.

Mack, D.; Bortfeldt, A.; Gehring, H. (2004): A parallel hybrid local search algorithm for the container loading problem. In: *International Transactions in Operational Research*, 11, 5, pp. 511–533.

Madsen, O. B. (1976): Optimal scheduling of trucks - A routing problem with tight due times for delivery. In: *Optimization applied to transportation systems*, pp. 126–136.

Männel, D.; Bortfeldt, A. (2016): A hybrid algorithm for the vehicle routing problem with pickup and delivery and three-dimensional loading constraints. In: *European Journal of Operational Research*, 254, 3, pp. 840–858.

Maquera, G.; Laguna, M.; Gandelman, D. A.; Sant'Anna, A. P. (2012): Scatter Search Applied to the Vehicle Routing Problem with Simultaneous Delivery and Pickup. In: P.-Y. Yin, ed., *Trends in Developing Metaheuristics, Algorithms, and Optimization Approaches*, pp. 149–168, IGI Global.

Martello, S.; Pisinger, D.; Vigo, D. (2000): The Three-Dimensional Bin Packing Problem. In: *Operations Research*, 48, 2, pp. 256–267.

Massen, F.; Deville, Y.; van Hentenryck, P. (2012): Pheromone-Based Heuristic Column Generation for Vehicle Routing Problems with Black Box Feasibility. In: N. Beldiceanu; N. Jussien; É. Pinson, eds., *Integration of AI and OR techniques in constraint programming for combinatorial optimization problems*, vol. 7298 of *SpringerLink Bücher*, pp. 260–274, Springer, Berlin.

Mester, D.; Bräysy, O.; Dullaert, W. (2007): A multi-parametric evolution strategies algorithm for vehicle routing problems. In: *Expert Systems with Applications*, 32, 2, pp. 508–517.

Miao, L.; Ruan, Q.; Woghiren, K.; Ruo, Q. (2012): A hybrid genetic algorithm for the vehicle routing problem with three-dimensional loading constraints. In: *RAIRO - Operations Research*, 46, 1, pp. 63–82.

Min, H. (1989): The multiple vehicle routing problem with simultaneous delivery and pick-up points. In: *Transportation Research Part A: General*, 23, 5, pp. 377–386.

Min, H.; Current, J.; Schilling, D. (1992): The multiple depot vehicle routing problem with backhauling. In: *Journal of Business Logistics*, 13, 1, pp. 259.

Mingozzi, A.; Giorgi, S.; Baldacci, R. (1999): An Exact Method for the Vehicle Routing Problem with Backhauls. In: *Transportation Science*, 33, 3, pp. 315–329.

Mitra, S. (2005): An Algorithm for the Generalized Vehicle Routing Problem With Backhauling. In: *Asia-Pacific Journal of Operational Research*, 22, 02, pp. 153–169.

Mosheiov, G. (1998): Vehicle routing with pick-up and delivery: Tour-partitioning heuristics. In: *Computers & Industrial Engineering*, 34, 3, pp. 669–684.

Moura, A. (2008): A Multi-Objective Genetic Algorithm for the Vehicle Routing with Time Windows and Loading Problem. In: A. Bortfeldt; J. Homberger; H. Kopfer; G. Pankratz; R. Strangmeier, eds., *Intelligent Decision Support*, Gabler Edition Wissenschaft, pp. 187–201, Betriebswirtschaftlicher Verlag Dr. Th. Gabler / GWV Fachverlage GmbH Wiesbaden, Wiesbaden.

Moura, A.; Oliveira, J. F. (2005): A GRASP Approach to the Container-Loading Problem. In: *IEEE Intelligent Systems*, 20, 4, pp. 50–57.

Moura, A.; Oliveira, J. F. (2009): An integrated approach to the vehicle routing and container loading problems. In: *OR Spectrum*, 31, 4, pp. 775–800.

Nagy, G.; Salhi, S. (2005): Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. In: *European Journal of Operational Research*, 162, 1, pp. 126–141.

Nagy, G.; Wassan, N. A.; Salhi, S. (2013): The vehicle routing problem with restricted mixing of deliveries and pickups. In: *Journal of Scheduling*, 16, 2, pp. 199–213.

Nagy, G.; Wassan, N. A.; Speranza, M. G.; Archetti, C. (2015): The Vehicle Routing Problem with Divisible Deliveries and Pickups. In: *Transportation Science*, 49, 2, pp. 271–294.

Ong, J. O.; Suprayogi (2011): Vehicle Routing Problem with Backhaul, Multiple Trips and Time Window. In: *Jurnal Teknik Industri*, 13, 1.

Osman, I. H.; Wassan, N. A. (2002): A reactive tabu search meta-heuristic for the vehicle routing problem with back-hauls. In: *Journal of Scheduling*, 5, 4, pp. 263–285.

Pace, S.; Turky, A.; Moser, I.; Aleti, A. (2015): Distributing Fibre Boards: A Practical Application of the Heterogeneous Fleet Vehicle Routing Problem with Time Windows and Three-dimensional Loading Constraints. In: *Procedia Computer Science*, 51, pp. 2257–2266.

Padberg, M. (2000): Packing small boxes into a big box. In: *Mathematical Methods of Operational Research*, 52, 1, pp. 1–21.

Palhazi Cuervo, D.; Goos, P.; Sörensen, K.; Arráiz, E. (2014): An iterated local search algorithm for the vehicle routing problem with backhauls. In: *European Journal of Operational Research*, 237, 2, pp. 454–464.

Parragh, S. N.; Doerner, K. F.; Hartl, R. F. (2008): A survey on pickup and delivery problems. In: *Journal für Betriebswirtschaft*, 58, 1, pp. 21–51.

Parreño, F.; Alvarez-Valdes, R.; Tamarit, J. M.; Oliveira, J. F. (2008): A Maximal-Space Algorithm for the Container Loading Problem. In: *INFORMS Journal on Computing*, 20, 3, pp. 412–422.

Pinto, T.; Alves, C.; De, C.; Moura, A. (2015): An insertion heuristic for the capacitated vehicle routing problem with loading constraints and mixed linehauls and backhauls. In: *FME Transaction*, 43, 4, pp. 311–318.

Pisinger, D. (2002): Heuristics for the container loading problem. In: *European Journal of Operational Research*, 141, 2, pp. 382–392.

Pisinger, D.; Ropke, S. (2007): A general heuristic for vehicle routing problems. In: *Computers & Operations Research*, 34, 8, pp. 2403–2435.

Polat, O.; Kalayci, C. B.; Kulak, O.; Günther, H.-O. (2015): A perturbation based variable neighborhood search heuristic for solving the Vehicle Routing Problem with Simultaneous Pickup and Delivery with Time Limit. In: *European Journal of Operational Research*, 242, 2, pp. 369–382.

Pollaris, H.; Braekers, K.; Caris, A.; Janssens, G. K.; Limbourg, S. (2015): Vehicle routing problems with loading constraints: State-of-the-art and future directions. In: *OR Spectrum*, 37, 2, pp. 297–330.

Potvin, J.-Y.; Duhamel, C.; Guertin, F. (1996): A genetic algorithm for vehicle routing with backhauling. In: *Applied Intelligence*, 6, 4, pp. 345–355.

Pullen, H. G. M. (1967): A computer application to a transport scheduling problem. In: *The Computer Journal*, 10, 1, pp. 10–13.

Ratcliff, M. S. W.; Bischoff, E. E. (1998): Allowing for weight considerations in container loading. In: *OR Spektrum*, 20, 1, pp. 65–71.

Reil, S.; Bortfeldt, A.; Mönch, L. (2017): Heuristics for vehicle routing problems with backhauls, time windows, and 3D loading constraints. In: *European Journal of Operational Research*.

Reimann, M.; Doerner, K.; Hartl, R. F. (2002): Insertion Based Ants for Vehicle Routing Problems with Backhauls and Time Windows. In: M. Dorigo; G. Di Caro; M. Sampels, eds., *Ant Algorithms: Third International Workshop, ANTS 2002 Brussels, Belgium, September 12–14, 2002 Proceedings*, pp. 135–148, Springer Berlin Heidelberg, Berlin, Heidelberg.

Reimann, M.; Ulrich, H. (2006): Comparing backhauling strategies in vehicle routing using Ant Colony Optimization. In: *Central European Journal of Operations Research*, 14, 2, pp. 105–123.

Reinelt, G. (1991): TSPLIB—A Traveling Salesman Problem Library. In: *ORSA Journal on Computing*, 3, 4, pp. 376–384.

Rieck, J.; Zimmermann, J. (2013): Exact Solutions to the Symmetric and Asymmetric Vehicle Routing Problem with Simultaneous Delivery and Pick-Up. In: *Business Research*, 6, 1, pp. 77–92.

Ropke, S.; Pisinger, D. (2006a): A unified heuristic for a large class of Vehicle Routing Problems with Backhauls. In: *European Journal of Operational Research*, 171, 3, pp. 750–775.

Ropke, S.; Pisinger, D. (2006b): An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. In: *Transportation Science*, 40, 4, pp. 455–472.

Ruan, Q.; Zhang, Z.; Miao, L.; Shen, H. (2013): A hybrid approach for the vehicle routing problem with three-dimensional loading constraints. In: *Computers & Operations Research*, 40, 6, pp. 1579–1589.

Salani, M. (2006): *Branch-and-price algorithms for vehicle routing problems*. *PhD Thesis*, Università degli studi di Milano.

Salhi, S.; Nagy, G. (1999): A Cluster Insertion Heuristic for Single and Multiple Depot Vehicle Routing Problems with Backhauling. In: *The Journal of the Operational Research Society*, 50, 10, pp. 1034.

Shaw, P. (1997): A new local search algorithm providing high quality solutions to vehicle routing problems. In: *APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, Working Paper*.

Shaw, P. (1998): Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. In: M. Maher; J.-F. Puget, eds., *Principles and Practice of Constraint Programming — CP98*, vol. 1520 of *Lecture Notes in Computer Science*, pp. 417–431, Springer Berlin Heidelberg, Berlin, Heidelberg.

Sixt, M. (1996): *Dreidimensionale Packprobleme: Lösungsverfahren basierend auf den Meta-Heuristiken Simulated Annealing und Tabu-Suche*. Peter Lang, Europäischer Verlag der Wissenschaften, Frankfurt am Main.

Solomon, M. M. (1987): Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. In: *Operations Research*, 35, 2, pp. 254–265.

Subramanian, A.; Drummond, L.; Bentes, C.; Ochi, L. S.; Farias, R. (2010a): A parallel heuristic for the Vehicle Routing Problem with Simultaneous Pickup and Delivery. In: *Computers & Operations Research*, 37, 11, pp. 1899–1911.

Subramanian, A.; Uchoa, E.; Ochi, L. S. (2010b): New Lower Bounds for the Vehicle Routing Problem with Simultaneous Pickup and Delivery. In: P. Festa, ed., *Experimental algorithms*, vol. 6049 of *Lecture Notes in Computer Science*, pp. 276–287, Springer, Berlin.

Subramanian, A.; Uchoa, E.; Pessoa, A. A.; Ochi, L. S. (2013): Branch-cut-and-price for the vehicle routing problem with simultaneous pickup and delivery. In: *Optimization Letters*, 7, 7, pp. 1569–1581.

Talbi, E.-G. (2009): *Metaheuristics: From design to implementation*, vol. v.74 of *Wiley Series on Parallel and Distributed Computing*. John Wiley & Sons, Hoboken, NJ.

Tang Montané, F. A.; Galvão, R. D. (2006): A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. In: *Computers & Operations Research*, 33, 3, pp. 595–619.

Tao, Y.; Wang, F. (2015): An effective tabu search approach with improved loading algorithms for the 3L-CVRP. In: *Computers & Operations Research*, 55, pp. 127–140.

Tarantilis, C. D.; Zachariadis, E. E.; Kiranoudis, C. T. (2009): A Hybrid Metaheuristic Algorithm for the Integrated Vehicle Routing and Three-Dimensional Container-Loading Problem. In: *IEEE Transactions on Intelligent Transportation Systems*, 10, 2, pp. 255–271.

Tasan, A. S.; Gen, M. (2012): A genetic algorithm based approach to vehicle routing

problem with simultaneous pick-up and deliveries. In: *Computers & Industrial Engineering*, 62, 3, pp. 755–761.

Terno, J.; Scheithauer, G.; Sommerweiß, U.; Riehme, J. (2000): An efficient approach for the multi-pallet loading problem. In: *European Journal of Operational Research*, 123, 2, pp. 372–381.

Thangiah, S. R.; Potvin, J.-Y.; Sun, T. (1996): Heuristic approaches to vehicle routing with backhauls and time windows. In: *Computers & Operations Research*, 23, 11, pp. 1043–1057.

Toth, P.; Vigo, D. (1996): A Heuristic Algorithm for the Vehicle Routing Problem with Backhauls. In: L. Bianco; P. Toth, eds., *Advanced Methods in Transportation Analysis*, pp. 585–608, Springer Berlin Heidelberg, Berlin, Heidelberg.

Toth, P.; Vigo, D. (1997): An Exact Algorithm for the Vehicle Routing Problem with Backhauls. In: *Transportation Science*, 31, 4, pp. 372–385.

Toth, P.; Vigo, D. (1999): A heuristic algorithm for the symmetric and asymmetric vehicle routing problems with backhauls. In: *European Journal of Operational Research*, 113, 3, pp. 528–543.

Tütüncü, G. Y.; Carreto, C. A. C.; Baker, B. M. (2009): A visual interactive approach to classical and mixed vehicle routing problems with backhauls. In: *Omega*, 37, 1, pp. 138–154.

Vidal, T.; Crainic, T. G.; Gendreau, M.; Prins, C. (2013): A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. In: *Computers & Operations Research*, 40, 1, pp. 475–489.

Wade, A.; Salhi, S. (2004): An Ant System Algorithm for the Mixed Vehicle Routing Problem with Backhauls. In: *Metaheuristics: Computer Decision-Making*, pp. 699–719, Springer US, Boston, MA.

Wade, A. C.; Salhi, S. (2002): An investigation into a new class of vehicle routing problem with backhauls. In: *Omega*, 30, 6, pp. 479–487.

Wang, C.; Mu, D.; Zhao, F.; Sutherland, J. W. (2015): A parallel simulated annealing method for the vehicle routing problem with simultaneous pickup–delivery and time windows. In: *Computers & Industrial Engineering*, 83, pp. 111–122.

Wang, H.-F.; Chen, Y.-Y. (2012): A genetic algorithm for the simultaneous delivery and pickup problems with time window. In: *Computers & Industrial Engineering*, 62, 1, pp. 84–95.

Wang, L.; Guo, S.; Chen, S.; Zhu, W.; Lim, A. (2010): Two Natural Heuristics for 3D Packing with Practical Loading Constraints. In: B.-T. Zhang; M. A. Orgun, eds., *PRICAI 2010: trends in artificial intelligence*, vol. 6230 of *Lecture notes in computer science Lecture notes in artificial intelligence*, pp. 256–267, Springer, Berlin.

Wäscher, G.; Haußner, H.; Schumann, H. (2007): An improved typology of cutting and packing problems. In: *European Journal of Operational Research*, 183, 3, pp. 1109–1130.

Wassan, N. (2007): Reactive tabu adaptive memory programming search for the vehicle routing problem with backhauls. In: *Journal of the Operational Research Society*, 58, 12, pp. 1630–1641.

Wassan, N. A.; Nagy, G.; Ahmadi, S. (2008a): A heuristic method for the vehicle routing problem with mixed deliveries and pickups. In: *Journal of Scheduling*, 11, 2, pp. 149–161.

Wassan, N. A.; Salhi, S.; Nagy, G.; Wassan, N.; Wade, A. C. (2013): Solving the Mixed Backhauling Vehicle Routing Problem with Ants. In: *International Journal of Energy Optimization and Engineering*, 2, 2, pp. 62–77.

Wassan, N. A.; Wassan, A. H.; Nagy, G. (2008b): A reactive tabu search algorithm for the vehicle routing problem with simultaneous pickups and deliveries. In: *Journal of Combinatorial Optimization*, 15, 4, pp. 368–386.

Wei, L.; Zhang, Z.; Lim, A. (2014): An Adaptive Variable Neighborhood Search for a Heterogeneous Fleet Vehicle Routing Problem with Three-Dimensional Loading Constraints. In: *IEEE Computational Intelligence Magazine*, 9, 4, pp. 18–30.

Wisniewski, M. A.; Ritt, M.; Buriol, L. S. (2011): A tabu search algorithm for the capacitated vehicle routing problem with three-dimensional loading constraints. In: *XLIII Simposio Brasilero de Pesquisa Operacional*.

Yano, C. A.; Chan, T. J.; Richter, L. K.; Cutler, T.; Murty, K. G.; McGettigan, D. (1987): Vehicle Routing at Quality Stores. In: *Interfaces*, 17, 2, pp. 52–63.

Zachariadis, E. E.; Kiranoudis, C. T. (2012): An effective local search approach for the Vehicle Routing Problem with Backhauls. In: *Expert Systems with Applications*, 39, 3, pp. 3174–3184.

Zachariadis, E. E.; Tarantilis, C. D.; Kiranoudis, C. T. (2009): A hybrid metaheuristic algorithm for the vehicle routing problem with simultaneous delivery and pick-up service. In: *Expert Systems with Applications*, 36, 2, pp. 1070–1081.

Zachariadis, E. E.; Tarantilis, C. D.; Kiranoudis, C. T. (2010): An adaptive memory methodology for the vehicle routing problem with simultaneous pick-ups and deliveries. In: *European Journal of Operational Research*, 202, 2, pp. 401–411.

Zachariadis, E. E.; Tarantilis, C. D.; Kiranoudis, C. T. (2012): The Pallet-Packing Vehicle Routing Problem. In: *Transportation Science*, 46, 3, pp. 341–358.

Zachariadis, E. E.; Tarantilis, C. D.; Kiranoudis, C. T. (2016): The Vehicle Routing Problem with Simultaneous Pick-ups and Deliveries and Two-Dimensional Loading Constraints. In: *European Journal of Operational Research*, 251, 2, pp. 369–386.

Zhang, D.; Cai, S.; Ye, F.; Si, Y.-W.; Nguyen, T. T. (2017): A hybrid algorithm for a vehicle routing problem with realistic constraints. In: *Information Sciences*, 394-395, pp. 167–182.

Zhang, T.; Chaovalitwongse, W. A.; Zhang, Y. (2012): Scatter search for the stochastic travel-time vehicle routing problem with simultaneous pick-ups and deliveries. In: *Computers & Operations Research*, 39, 10, pp. 2277–2290.

Zhang, T.; Chaovalitwongse, W. A.; Zhang, Y. (2014): Integrated Ant Colony and Tabu Search approach for time dependent vehicle routing problems with simultaneous pickup and delivery. In: *Journal of Combinatorial Optimization*, 28, 1, pp. 288–309.

Zhang, Z.; Wei, L.; Lim, A. (2015): An evolutionary local search for the capacitated vehicle routing problem minimizing fuel consumption under three-dimensional loading constraints. In: *Transportation Research Part B: Methodological*, 82, pp. 20–35.

Zhao, X.; Bennell, J. A.; Bektaş, T.; Dowsland, K. (2016): A comparative review of 3D container loading algorithms. In: *International Transactions in Operational Research*, 23, 1-2, pp. 287–320.

Zheng, J.-N.; Chien, C.-F.; Gen, M. (2015): Multi-objective multi-population biased random-key genetic algorithm for the 3-D container loading problem. In: *Computers & Industrial Engineering*, 89, pp. 80–87.

Zhong, Y.; Cole, M. H. (2005): A vehicle routing problem with backhauls and time windows: A guided local search solution. In: *Transportation Research Part E: Logistics and Transportation Review*, 41, 2, pp. 131–144.

Zhu, W.; Qin, H.; Lim, A.; Wang, L. (2012): A two-stage tabu search algorithm with enhanced packing heuristics for the 3L-CVRP and M3L-CVRP. In: *Computers & Operations Research*, 39, 9, pp. 2178–2195.

Appendix

Appendix A

Removal heuristics

In order to determine the most suitable set of removal heuristics, 16 different sets were defined. In Table A.1, the removal heuristics included in each set are listed. Set 1 contains all removal heuristics considered in this thesis (see Chapter 4.2.2.1). Set 2 contains those proposed by Ropke and Pisinger (2006a,b). Set 3 contains allegedly good heuristics with large average weights throughout the ALNS procedure (recorded in pre-tests) indicating that they produce good solutions. In sets 4 and 5 the route removal heuristics and the newly developed heuristics, respectively, are excluded. The remaining sets are based on set 2. The new removal heuristics replace single heuristics of Ropke and Pisinger (2006a,b).

Table A.1: Explanation of tested sets of removal heuristics

removal heuristic	**set** 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Shaw	✓	✓		✓	✓	✓	✓	✓	✓	✓				✓	✓	✓
random	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
worst	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
cluster	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
neighb. graph	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			
worst dist.	✓		✓	✓	✓											
worst time	✓			✓	✓											
Shaw proximity	✓			✓	✓											
Shaw time	✓			✓	✓											
Shaw demand	✓			✓	✓											
historical knowl.	✓		✓	✓	✓											
avg. dist.	✓		✓	✓	✓											
node neighb.	✓			✓	✓											
overlap	✓			✓		✓	✓		✓	✓		✓	✓		✓	✓
random-route	✓				✓											
least cust.-route	✓					✓										
avg. dist.-route	✓					✓										
largest dist.-route	✓					✓										
inner route	✓			✓		✓	✓									
intersect	✓		✓	✓		✓	✓	✓		✓	✓		✓	✓		✓
route pair	✓					✓	✓									

avg.: average, cust.: customers, dist.: distance, knowl.: knowledge, neighb.: neighbourhood

The ALNS is applied to 50 test instances of different VRPBTW variants. In Table A.2, the results of these experiments are presented. It contains the average deviations (∅dev) of the TTDs of the obtained solutions from those of the best

known solutions from the literature. In addition, numbers of instances are provided for which the respective heuristic set found the best solution among the considered sets (#best).

Table A.2: Comparison of sets of removal heuristics; average deviation from best known solutions and number of best solutions (50 test instances)

set	**∅dev[%]**	**#best**
1	0.170	**37**
2	0.210	33
3	0.621	27
4	0.191	34
5	0.171	32
6	0.222	31
7	0.191	**37**
8	0.377	34
9	0.266	31
10	0.387	32
11	0.250	30
12	**0.162**	33
13	0.167	31
14	0.446	25
15	0.300	30
16	0.397	26

#best: number of best solutions (among the considered sets), ∅dev: average deviation from best known solution.
Minimum deviation and maximum numbers of best solutions are bold-faced.

Appendix B

Detailed results for VRPBTW instances

In this section, the detailed results of applying the ALNS to the 1D instance sets described in Chapter 5.2.1 are provided. Abbreviations are used for the references. They are listed in Table B.1.

Table B.1: Reference abbreviations

abbreviation	**reference**
G89	Goetschalckx and Jacobs-Blecha (1989)
G95	Gélinas et al. (1995)
P96	Potvin et al. (1996)
T97	Toth and Vigo (1997)
C99	Cook and Rich (1999)
K99	Kohl et al. (1999)
L99	Larsen (1999)
K00	Kallehauge et al. (2000)
O02	Osman and Wassan (2002)
R02	Reimann et al. (2002)
B06	Brandão (2006)
C06	Chabrier (2006)
I06	Irnich and Villeneuve (2006)
R06	Ropke and Pisinger (2006a)
S06	Salani (2006)
T06	Tang Montané and Galvão (2006)
M07	Mester et al. (2007)
D08	Desaulniers et al. (2008)
J08	Jepsen et al. (2008)
W08	Wassan et al. (2008b)
G09	Gajpal and Abad (2009)
Z09	Zachariadis et al. (2009)
S10	Subramanian et al. (2010a)
Z10	Zachariadis et al. (2010)
B11	Baldacci et al. (2011)
J12	Jun and Kim (2012)
G15	García-Nájera et al. (2015)

The results for the VRPBTW instances are given in Table B.2. Information is provided about the instance sets, instance names, problem variants and the numbers of customers (n). The instance sets refer to those introduced in Chapter 5.2.1 (cf. Table 5.1). Furthermore, the TTDs per instance of the best known solutions (BKS), the corresponding references (ref) presenting the BKS for the first time and the best TTDs provided by the ALNS are given as well as their deviations (dev) from the BKS.

Table B.2: ALNS results for VRPBTW instances

				BKS		**ALNS**	
				TTD	ref	TTD	dev
set	**instance**	**problem**	n	[DU]		[DU]	[%]
Sol87	C101	VRPTW	100	827.3	K99	828.9	0.20
Sol87	C102	VRPTW	100	827.3	K99	828.9	0.20
Sol87	C103	VRPTW	100	826.3	K99	828.1	0.21
Sol87	C104	VRPTW	100	822.9	K99	824.8	0.23
Sol87	C105	VRPTW	100	827.3	K99	828.9	0.20
Sol87	C106	VRPTW	100	827.3	K99	828.9	0.20
Sol87	C107	VRPTW	100	827.3	K99	828.9	0.20
Sol87	C108	VRPTW	100	827.3	K99	828.9	0.20
Sol87	C109	VRPTW	100	827.3	K99	828.9	0.20
Sol87	C201	VRPTW	100	589.1	C99	591.6	0.42
Sol87	C202	VRPTW	100	589.1	C99	591.6	0.42
Sol87	C203	VRPTW	100	588.7	K00	591.2	0.42
Sol87	C204	VRPTW	100	588.1	I06	590.6	0.42
Sol87	C205	VRPTW	100	586.4	C99	588.9	0.42
Sol87	C206	VRPTW	100	586.0	C99	588.5	0.43
Sol87	C207	VRPTW	100	585.8	C99	588.3	0.42
Sol87	C208	VRPTW	100	585.8	K00	588.3	0.43
Sol87	R101	VRPTW	100	1,637.7	K99	1,642.9	0.32
Sol87	R102	VRPTW	100	1,466.6	K99	1,473.6	0.48
Sol87	R103	VRPTW	100	1,208.7	C99,L99	1,213.6	0.41
Sol87	R104	VRPTW	100	971.5	I06	981.2	1.00
Sol87	R105	VRPTW	100	1,355.3	K99	1,360.8	0.40
Sol87	R106	VRPTW	100	1,234.6	C99	1,239.4	0.39
Sol87	R107	VRPTW	100	1,064.6	C99	1,072.1	0.71
Sol87	R108	VRPTW	100	932.1	J08	938.2	0.65
Sol87	R109	VRPTW	100	1,146.9	C99	1,151.8	0.43
Sol87	R110	VRPTW	100	1,068.0	C99	1,072.4	0.41
Sol87	R111	VRPTW	100	1,048.7	C99	1,053.5	0.46
Sol87	R112	VRPTW	100	948.6	J08	955.7	0.75
Sol87	R201	VRPTW	100	1,143.2	K00	1,152.3	0.79
Sol87	R202	VRPTW	100	1,029.6	J08	1,038.5	0.87
Sol87	R203	VRPTW	100	870.8	J08	876.3	0.63
Sol87	R204	VRPTW	100	731.3	B11	735.8	0.61
Sol87	R205	VRPTW	100	949.8	D08	954.2	0.46
Sol87	R206	VRPTW	100	875.9	D08	884.8	1.02
Sol87	R207	VRPTW	100	794.0	D08	798.0	0.50
Sol87	R208	VRPTW	100	725.8	M07	706.9	−2.60
Sol87	R209	VRPTW	100	854.8	J08	860.5	0.67
Sol87	R210	VRPTW	100	900.5	D08	912.5	1.33
Sol87	R211	VRPTW	100	746.7	B11	757.6	1.47
Sol87	RC101	VRPTW	100	1,619.8	K99	1,627.7	0.49
Sol87	RC102	VRPTW	100	1,457.4	C99	1,461.2	0.26
Sol87	RC103	VRPTW	100	1,258.0	C99	1,272.0	1.11
Sol87	RC104	VRPTW	100	1,132.3	I06	1,135.8	0.31
Sol87	RC105	VRPTW	100	1,513.7	K99	1,518.6	0.32
Sol87	RC106	VRPTW	100	1,372.7	S06	1,379.6	0.50
Sol87	RC107	VRPTW	100	1,207.8	I06	1,212.8	0.42
Sol87	RC108	VRPTW	100	1,114.2	I06	1,118.1	0.35

Table B.2: Results ALNS for VRPBTW instances (*continued*)

				BKS		**ALNS**	
				TTD	ref	TTD	dev
set	**instance**	**problem**	n	[DU]		[DU]	[%]
Sol87	RC201	VRPTW	100	1,261.8	K00	1,265.6	0.30
Sol87	RC202	VRPTW	100	1,092.3	I06,C06	1,098.8	0.60
Sol87	RC203	VRPTW	100	923.7	J08	939.7	1.73
Sol87	RC204	VRPTW	100	783.5	B11	787.5	0.51
Sol87	RC205	VRPTW	100	1,154.0	I06,C06	1,157.6	0.31
Sol87	RC206	VRPTW	100	1,051.1	J08	1,061.4	0.98
Sol87	RC207	VRPTW	100	962.9	D08	969.6	0.69
Sol87	RC208	VRPTW	100	776.1	B11	791.4	1.97
GJB89	A1	VRPCB	25	229,886.0	T97	229,886.0	0.00
GJB89	A2	VRPCB	25	180,119.0	T97	180,119.0	0.00
GJB89	A3	VRPCB	25	155,796.4	G15	155,796.0	0.00
GJB89	A4	VRPCB	25	155,796.0	T97	155,796.0	0.00
GJB89	B1	VRPCB	30	239,080.0	T97	239,080.0	0.00
GJB89	B2	VRPCB	30	198,048.0	T97	198,048.0	0.00
GJB89	B3	VRPCB	30	169,372.0	G89	169,372.0	0.00
GJB89	C1	VRPCB	40	249,448.0	T97	250,557.0	0.44
GJB89	C2	VRPCB	40	215,020.0	T97	215,020.0	0.00
GJB89	C3	VRPCB	40	195,366.6	G15	195,367.0	0.00
GJB89	C4	VRPCB	40	195,366.0	T97	195,367.0	0.00
GJB89	D1	VRPCB	38	317,491.6	G15	316,709.0	−0.25
GJB89	D2	VRPCB	38	316,708.9	T97	316,709.0	0.00
GJB89	D3	VRPCB	38	239,478.6	T97	239,479.0	0.00
GJB89	D4	VRPCB	38	205,831.9	T97	205,832.0	0.00
GJB89	E1	VRPCB	45	238,879.6	T97	238,880.0	0.00
GJB89	E2	VRPCB	45	212,263.0	T97	212,263.0	0.00
GJB89	E3	VRPCB	45	206,659.0	T97	206,659.0	0.00
GJB89	F1	VRPCB	60	263,173.0	T97	263,174.0	0.00
GJB89	F2	VRPCB	60	265,213.0	T97	263,174.0	−0.77
GJB89	F3	VRPCB	60	241,120.0	T97	241,970.0	0.35
GJB89	F4	VRPCB	60	233,861.0	T97	233,862.0	0.00
GJB89	G1	VRPCB	57	306,305.0	O02	305,002.0	−0.43
GJB89	G2	VRPCB	57	245,441.0	T97	245,441.0	0.00
GJB89	G3	VRPCB	57	229,507.0	T97	229,507.0	0.00
GJB89	G4	VRPCB	57	229,507.5	G15	229,507.0	0.00
GJB89	G5	VRPCB	57	218,716.6	G15	218,485.0	−0.11
GJB89	G6	VRPCB	57	213,457.0	T97	213,457.0	0.00
GJB89	H1	VRPCB	68	268,933.0	T97	268,933.0	0.00
GJB89	H2	VRPCB	68	253,365.0	T97	253,365.0	0.00
GJB89	H3	VRPCB	68	247,449.0	T97	247,449.0	0.00
GJB89	H4	VRPCB	68	247,449.1	G15	247,449.0	0.00
GJB89	H5	VRPCB	68	246,121.0	T97	246,121.0	0.00
GJB89	H6	VRPCB	68	246,121.3	G15	246,604.0	0.20
GJB89	I1	VRPCB	90	347,832.7	G15	347,833.0	0.00
GJB89	I2	VRPCB	90	309,943.0	R06	309,944.0	0.00
GJB89	I3	VRPCB	90	294,507.0	O02	294,507.0	0.00
GJB89	I4	VRPCB	90	294,834.0	G15	294,507.0	−0.11
GJB89	I5	VRPCB	90	294,868.3	G15	294,507.0	−0.12
GJB89	J1	VRPCB	94	335,006.0	R06	335,480.0	0.14

Table B.2: Results ALNS for VRPBTW instances (*continued*)

				BKS		**ALNS**	
				TTD	ref	TTD	dev
set	**instance**	**problem**	n	[DU]		[DU]	[%]
GJB89	J2	VRPCB	94	310,417.0	R06	310,417.0	0.00
GJB89	J3	VRPCB	94	279,219.0	O02	279,219.0	0.00
GJB89	J4	VRPCB	94	296,533.0	R06	296,959.0	0.14
GJB89	K1	VRPCB	113	394,071.2	G09	394,376.0	0.08
GJB89	K2	VRPCB	113	362,130.0	R06	362,130.0	0.00
GJB89	K3	VRPCB	113	364,086.7	G15	362,130.0	−0.54
GJB89	K4	VRPCB	113	348,949.4	R06	348,949.0	0.00
GJB89	L1	VRPCB	150	417,896.7	G09	426,176.0	1.98
GJB89	L2	VRPCB	150	401,228.0	R06	401,466.0	0.06
GJB89	L3	VRPCB	150	402,677.7	R06	401,466.0	−0.30
GJB89	L4	VRPCB	150	384,636.3	R06	384,636.0	0.00
GJB89	L5	VRPCB	150	387,564.6	R06	384,636.0	−0.76
GJB89	M1	VRPCB	125	398,593.0	G09	398,497.0	−0.02
GJB89	M2	VRPCB	125	396,917.0	G09	398,093.0	0.30
GJB89	M3	VRPCB	125	375,695.4	G09	376,159.0	0.12
GJB89	M4	VRPCB	125	348,140.2	G09	348,533.0	0.11
GJB89	N1	VRPCB	150	408,100.6	G09	412,061.0	0.97
GJB89	N2	VRPCB	150	408,065.4	G09	409,746.0	0.41
GJB89	N3	VRPCB	150	394,337.9	R06	397,603.0	0.83
GJB89	N4	VRPCB	150	394,788.4	R06	399,551.0	1.21
GJB89	N5	VRPCB	150	373,476.3	R06	375,347.0	0.50
GJB89	N6	VRPCB	150	373,758.7	R06	375,871.0	0.57
TV97	eil22_50	VRPCB	21	371.0	T97	372.1	0.30
TV97	eil22_66	VRPCB	21	366.0	T97	367.0	0.28
TV97	eil22_80	VRPCB	21	375.0	T97	376.9	0.51
TV97	eil23_50	VRPCB	22	682.0	T97	682.3	0.05
TV97	eil23_66	VRPCB	22	649.0	T97	648.4	−0.09
TV97	eil23_80	VRPCB	22	623.0	T97	623.6	0.09
TV97	eil30_50	VRPCB	29	501.0	T97	502.8	0.35
TV97	eil30_66	VRPCB	29	537.0	T97	537.7	0.13
TV97	eil30_80	VRPCB	29	514.0	T97	517.8	0.75
TV97	eil33_50	VRPCB	32	738.0	T97	738.8	0.10
TV97	eil33_66	VRPCB	32	750.0	T97	751.1	0.15
TV97	eil33_80	VRPCB	32	736.0	T97	737.4	0.19
TV97	eil51_50	VRPCB	50	559.0	T97	562.8	0.69
TV97	eil51_66	VRPCB	50	548.0	T97	550.9	0.52
TV97	eil51_80	VRPCB	50	565.0	T97	569.5	0.80
TV97	eilA101_50	VRPCB	100	831.0	R06	845.9	1.80
TV97	eilA101_66	VRPCB	100	846.0	T97	858.3	1.46
TV97	eilA101_80	VRPCB	100	857.0	R06	870.6	1.59
TV97	eilA76_50	VRPCB	75	739.0	T97	743.6	0.62
TV97	eilA76_66	VRPCB	75	768.0	T97	771.8	0.50
TV97	eilA76_80	VRPCB	75	781.0	T97	794.2	1.69
TV97	eilB101_50	VRPCB	100	925.0	R06,B06	940.8	1.71
TV97	eilB101_66	VRPCB	100	989.0	R06	1,008.6	1.98
TV97	eilB101_80	VRPCB	100	1,008.0	R06,B06	1,022.2	1.41
TV97	eilB76_50	VRPCB	75	801.0	T97	805.6	0.58
TV97	eilB76_66	VRPCB	75	873.0	T97	875.5	0.29

Table B.2: Results ALNS for VRPBTW instances (*continued*)

				BKS		**ALNS**	
set	**instance**	**problem**	n	TTD [DU]	ref	TTD [DU]	dev [%]
TV97	eilB76_80	VRPCB	75	919.0	T97	925.5	0.71
TV97	eilC76_50	VRPCB	75	713.0	T97	718.0	0.71
TV97	eilC76_66	VRPCB	75	734.0	T97	739.7	0.78
TV97	eilC76_80	VRPCB	75	733.0	T97	742.4	1.28
TV97	eilD76_50	VRPCB	75	690.0	T97	695.3	0.77
TV97	eilD76_66	VRPCB	75	715.0	T97	719.9	0.68
TV97	eilD76_80	VRPCB	75	694.0	R06,B06	705.3	1.63
GDDS95	BHR101A	VRPCBTW	100	1,767.9	G95	1,810.5	2.41
GDDS95	BHR101B	VRPCBTW	100	1,877.6	G95	1,923.9	2.47
GDDS95	BHR101C	VRPCBTW	100	1,895.1	G95	1,930.2	1.85
GDDS95	BHR102A	VRPCBTW	100	1,600.5	G95	1,620.6	1.26
GDDS95	BHR102B	VRPCBTW	100	1,639.2	G95	1,750.7	6.80
GDDS95	BHR102C	VRPCBTW	100	1,721.3	G95	1,774.5	3.09
GDDS95	BHR103A	VRPCBTW	100	1,334.7	P96	1,339.8	0.38
GDDS95	BHR103B	VRPCBTW	100	1,381.6	P96	1,375.6	−0.44
GDDS95	BHR103C	VRPCBTW	100	1,456.5	R06	1,455.8	−0.04
GDDS95	BHR104A	VRPCBTW	100	1,084.2	R06	1,084.2	0.00
GDDS95	BHR104B	VRPCBTW	100	1,128.3	R02	1,124.1	−0.37
GDDS95	BHR104C	VRPCBTW	100	1,191.4	R06	1,171.0	−1.71
GDDS95	BHR105A	VRPCBTW	100	1,544.8	R02	1,500.2	−2.89
GDDS95	BHR105B	VRPCBTW	100	1,583.3	R06	1,581.8	−0.10
GDDS95	BHR105C	VRPCBTW	100	1,633.0	R02	1,602.8	−1.85
SN99a	CMT01H	VRPMB	50	462.0	J12	465.1	0.66
SN99a	CMT01Q	VRPMB	50	490.0	R06	489.7	−0.05
SN99a	CMT01T	VRPMB	50	520.0	R06	520.1	0.01
SN99a	CMT02H	VRPMB	75	661.0	J12	662.6	0.25
SN99a	CMT02Q	VRPMB	75	732.0	R06	732.8	0.11
SN99a	CMT02T	VRPMB	75	783.0	R06	782.8	−0.03
SN99a	CMT03H	VRPMB	100	701.0	R06	700.9	−0.01
SN99a	CMT03Q	VRPMB	100	747.0	R06	747.2	0.02
SN99a	CMT03T	VRPMB	100	798.0	R06	803.0	0.63
SN99a	CMT04H	VRPMB	150	829.0	R06	830.9	0.23
SN99a	CMT04Q	VRPMB	150	915.0	J12	918.7	0.40
SN99a	CMT04T	VRPMB	150	993.0	G15	998.6	0.56
SN99a	CMT05H	VRPMB	199	983.0	R06	988.2	0.53
SN99a	CMT05Q	VRPMB	199	1,118.0	R06	1,113.4	−0.41
SN99a	CMT05T	VRPMB	199	1,227.0	R06	1,227.8	0.06
SN99a	CMT11H	VRPMB	120	818.0	R06	818.0	0.01
SN99a	CMT11Q	VRPMB	120	939.0	R06	939.4	0.04
SN99a	CMT11T	VRPMB	120	999.0	R06	998.8	−0.02
SN99a	CMT12H	VRPMB	100	629.0	R06	629.4	0.06
SN99a	CMT12Q	VRPMB	100	729.0	R06	729.2	0.03
SN99a	CMT12T	VRPMB	100	788.0	R06	787.5	−0.06
SN99b	CMT01X	VRPSDP	50	466.8	G15	470.5	0.80
SN99b	CMT01Y	VRPSDP	50	459.0	W08	461.2	0.50
SN99b	CMT02X	VRPSDP	75	668.8	W08	684.3	2.32
SN99b	CMT02Y	VRPSDP	75	663.3	W08	659.1	−0.63
SN99b	CMT03X	VRPSDP	100	715.3	J12	724.4	1.27

Table B.2: Results ALNS for VRPBTW instances (*continued*)

				BKS		**ALNS**	
set	**instance**	**problem**	n	TTD [DU]	ref	TTD [DU]	dev [%]
SN99b	CMT03Y	VRPSDP	100	719.0	T06	716.6	−0.33
SN99b	CMT04X	VRPSDP	150	852.5	Z09	855.5	0.36
SN99b	CMT04Y	VRPSDP	150	847.6	J12	827.7	−2.35
SN99b	CMT05X	VRPSDP	199	1,029.3	S10	1,030.5	0.12
SN99b	CMT05Y	VRPSDP	199	1,029.3	S10	978.9	−4.89
SN99b	CMT11X	VRPSDP	120	833.9	S10,Z10	829.8	−0.49
SN99b	CMT11Y	VRPSDP	120	830.4	W08	778.9	−6.21
SN99b	CMT12X	VRPSDP	100	644.7	W08	660.8	2.50
SN99b	CMT12Y	VRPSDP	100	659.5	W08	628.1	−4.76
total				90,061.7		90,167.2	0.30

Appendix C

Detailed results 3L-VRPBTW instances

In this section, the detailed results of applying the hybrid algorithms to the 3L-VRPBTW instances described in Chapter 5.2.2 are provided. For the analyses of the results, only those instances are considered for which a feasible solution is obtained at least once (within the conducted runs) by all problem and procedure settings that are compared (cf. p. 189). In Table C.1, the numbers of sufficiently solved instances are listed that are subject to the analyses in Chapter 5.5.2.1 and Chapter 5.5.2.2. Thus, they concern only instances with the constraint set C1. The numbers in the columns for the individual loading approaches ("RL", "LSP", "SL") refer to the instances used for comparing the hybrid algorithms for the individual loading approaches.

In order to compare loading approaches, instances are compared that are derived from the same core instances and differ only in the loading approach. That is, a core instance (or pair of instances) is taken into consideration here if both derived instances are solved feasibly. The corresponding numbers of core instances are listed in the column "all load".

Table C.1: Number of feasibly solved instances per loading approach and for all loading approaches (instances with C1 only)

problem	**no. of sufficiently solved (core) instances** RL	LSP	SL	all load	**total no. of instances**
with time windows					
3L-VRPTW	565		543	540	600
3L-VRPCBTW	567		534	531	600
3L-VRPMBTW		232	300	232	300
3L-VRPSDPTW		287	299	286	300
without time windows					
3L-VRPCB	567		548	545	600
3L-VRPMB		232	300	244	300
3L-VRPSDP		287	299	289	300

LSP: loading space partition, RL: rear loading, SL: side loading

For Chapters 5.5.2.3 to 5.5.2.5, all core instances are regarded as at least one instance is solved feasibly for each core instance, which is sufficient for these analyses.

In Chapter 5.5.3, instances are compared that are derived from the same core instances and differ only in the constraint set. Per packing heuristic and loading

approach, a core instance and its corresponding instances with the same loading approach are only regarded if all five instances (one per constraint set) are solved feasibly. The respective numbers are presented in Table C.2.

Table C.2: Add caption

problem variant	**no. of sufficiently solved instances** DBLF RL	DBLF LSP	DBLF SL	LS_DBLF RL	LS_DBLF LSP	LS_DBLF SL	**total no. of instances**
with time windows							
3L-VRPTW	162		156	216		208	240
3L-VRPCBTW	161		145	221		220	240
3L-VRPMBTW		54	117		95	120	120
3L-VRPSDPTW		106	119		118	119	
without time windows							
3L-VRPCB	170		157	223		222	240
3L-VRPMB		59	118		95	120	120
3L-VRPSDP		110	119		119	119	120

The abbreviations and symbols used in the following tables in this section are provided in Table C.3.

Table C.3: Abbreviations and symbols used in the tables in Appendix C

abbreviation/ symbol	**meaning**
-	no time windows
%LH	share of linehaul customers
*	benchmark
∅	average
ct	computing time
dev	deviation from benchmark TTD
DU	distance units
it.	item size
lrg.	large
m	total number of items
n	number of customers
nrr.	narrow
sm.	small
TTD	total travel distance
TW	time window width
typ	number of item types
wd.	wide

In Tables C.4 to C.7, the comprehensive results concerning the comparison of the variants of the hybrid algorithm (with respect to the employed packing heuristic) are presented. For each packing heuristic and loading approach, the average TTDs of the best solutions per instance (∅TTD), the average deviations of the obtained TTDs from the benchmarks (∅dev) and the average computing times per instance

and run (⌀ct) are given. As the deviations of the TTDs obtained by the benchmark procedure (ALNS×DBLF) naturally amount to 0 %, those deviations are omitted in the tables below. The results are provided for each problem variant separately and for the instances classes differing in the total number of items (if applicable[48]), the number of customers, share of linehaul customers or items (if applicable), time window width, item size and number of item types. Furthermore, the results are presented for two loading approaches separately. If the side loading is applied to the 3L-VRPMB(TW) and 3L-VRPSDP(TW), the variants DBLF^{SL} and $\text{LS\_DBLF}^{\text{SL}}$ are taken into account. Only those instances are considered that could be solved feasibly at least once for the respective loading approach by *all* alternative versions of the hybrid algorithm. Therefore, it may happen that some instance classes do not appear in the tables as no instance of that class was solved feasibly by all variants of the solution approach.

In Tables C.8 to C.11, the comprehensive results concerning the comparison of the loading approaches are presented. The best solutions per instance and packing heuristic obtained by rear loading (loading space partition) serve as benchmarks if linehaul and backhaul items are transported separately (simultaneously). Only those instances are considered that could be solved feasibly at least once by each packing heuristic and with both regarded loading approaches.

[48] This criterion is omitted in the case of the 3L-VRPMB(TW) and 3L-VRPSDP(TW) as only instances with 200 items are regarded for those problem variants.

Table C.4: Comparison of hybrid ALNS algorithms with different packing heuristics; extended problem variants (3L-VRPTW, RL/SL, C1); average TTDs, deviations from benchmark and computing times; separated by instance classes, loading approaches and variants of the hybrid algorithm

					ALNS														
					rear loading							**side loading**							
instance parameters					DBLF		LS_DBLF			LS_OS			DBLF		LS_DBLF			LS_OS	
					⌀TTD*	⌀ct	⌀TTD	⌀dev	⌀ct	⌀TTD	⌀dev	⌀ct	⌀TTD*	⌀ct	⌀TTD	⌀dev	⌀ct	⌀TTD	⌀dev	⌀ct
m	n	TW	it.	typ	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]
200	20	nrr.	sm.	3	439.8	6	438.4	−0.34	53	438.4	−0.34	31	440.7	4	438.6	−0.47	67	438.4	−0.52	10
200	20	nrr.	sm.	10	468.2	5	468.2	0.00	5	468.2	0.00	15	468.2	3	468.2	0.00	4	468.2	0.00	24
200	20	nrr.	sm.	100	450.1	12	447.7	−0.64	131	447.7	−0.64	294	450.7	10	447.7	−0.81	119	447.7	−0.81	309
200	20	wd.	sm.	3	391.5	178	389.6	−0.52	336	389.8	−0.48	301	391.7	124	389.8	−0.52	354	390.4	−0.34	347
200	20	wd.	sm.	10	375.6	278	367.6	−2.14	900	368.2	−1.98	900	378.7	234	367.7	−2.85	900	368.4	−2.69	900
200	20	wd.	sm.	100	411.2	125	397.1	−3.35	900	400.4	−2.53	900	413.7	118	396.8	−4.04	898	400.9	−3.01	900
200	60	nrr.	sm.	3	919.4	28	919.4	0.00	29	919.4	0.00	54	919.4	18	919.4	0.00	19	919.4	0.00	49
200	60	nrr.	sm.	10	961.6	32	961.6	0.00	33	961.6	0.00	105	961.6	18	961.6	0.00	18	961.6	0.00	126
200	60	nrr.	sm.	100	950.1	36	950.3	0.03	38	950.1	0.00	160	950.1	19	950.3	0.03	20	950.1	0.00	218
200	60	nrr.	lrg.	3	1,533.9	22	1,390.9	−8.90	203	1,393.8	−8.64	287	1,497.3	19	1,394.1	−6.80	173	1,395.0	−6.80	470
200	60	nrr.	lrg.	10	1,503.8	30	1,272.5	−15.08	566	1,265.9	−15.47	1,199	1,470.5	29	1,253.5	−14.36	608	1,245.4	−14.88	1,533
200	60	nrr.	lrg.	100	1,546.7	30	1,267.4	−17.97	695	1,251.2	−19.01	1,660	1,517.8	27	1,265.6	−16.52	660	1,245.9	−17.84	2,116
200	60	wd.	sm.	3	656.2	2,176	656.9	0.12	2,429	655.8	−0.06	2,319	657.0	1,722	656.3	−0.10	2,344	656.0	−0.16	2,540
200	60	wd.	sm.	10	650.6	2,476	648.2	−0.41	3,034	650.5	−0.02	3,523	650.0	1,818	649.3	−0.10	2,954	651.2	0.23	3,589
200	60	wd.	sm.	100	682.1	2,594	682.1	0.03	3,536	691.0	1.36	3,600	684.1	1,829	683.3	−0.19	3,484	695.7	1.63	3,600
200	60	wd.	lrg.	3	1,294.4	76	1,189.3	−7.48	1,664	1,182.9	−7.88	1,898	1,279.4	83	1,190.0	−6.72	1,771	1,189.3	−6.74	2,218
200	60	wd.	lrg.	10	1,426.3	78	1,225.9	−13.76	1,902	1,225.8	−13.71	2,849	1,416.0	76	1,222.8	−13.39	1,825	1,223.6	−13.23	3,001
200	60	wd.	lrg.	100	1,543.6	54	1,283.5	−16.78	1,508	1,263.5	−18.08	3,288	1,526.1	52	1,279.2	−16.14	1,480	1,253.7	−17.75	3,516
200	100	nrr.	sm.	3	1,265.4	78	1,265.4	0.00	77	1,265.4	0.00	131	1,265.4	53	1,265.4	0.00	65	1,265.4	0.00	130
200	100	nrr.	sm.	10	1,214.1	88	1,213.9	−0.02	93	1,214.2	0.00	269	1,214.1	54	1,214.1	0.00	56	1,214.2	0.00	301

Table C.4: Comparison of hybrid ALNS algorithms with different packing heuristics; extended problem variants (3L-VRPTW, RL/SL, C1); average TTDs, deviations from benchmark and computing times; separated by instance classes, loading approaches and variants of the hybrid algorithm (*continued*)

					ALNS														
					rear loading							**side loading**							
instance parameters					DBLF		LS_DBLF			LS_OS			DBLF		LS_DBLF			LS_OS	
					⌀TTD*	⌀ct	⌀TTD	⌀dev	⌀ct	⌀TTD	⌀dev	⌀ct	⌀TTD*	⌀ct	⌀TTD	⌀dev	⌀ct	⌀TTD	⌀dev	⌀ct
m	n	TW	it.	typ	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]
200	100	nrr.	sm.	100	1,289.0	89	1,288.8	−0.01	92	1,288.7	−0.02	318	1,289.2	61	1,287.2	−0.16	55	1,288.7	−0.04	378
200	100	nrr.	lrg.	3	1,523.5	87	1,470.8	−3.47	566	1,471.5	−3.40	981	1,554.9	78	1,484.0	−4.46	554	1,472.4	−5.20	1,228
200	100	nrr.	lrg.	10	1,675.3	116	1,524.1	−8.87	904	1,518.3	−9.21	2,392	1,632.2	113	1,517.7	−6.98	732	1,518.8	−6.87	2,617
200	100	nrr.	lrg.	100	1,774.9	92	1,592.6	−10.21	664	1,573.4	−11.29	2,149	1,756.6	75	1,588.1	−9.61	689	1,563.9	−10.98	2,463
200	100	wd.	sm.	3	882.9	2,655	882.1	−0.09	2,744	882.0	−0.11	2,338	882.2	2,153	882.5	0.04	2,128	882.1	−0.01	2,662
200	100	wd.	sm.	10	876.5	3,466	876.1	−0.04	3,453	880.8	0.51	3,600	877.7	2,886	877.4	−0.03	3,011	881.0	0.34	3,600
200	100	wd.	sm.	100	936.2	3,363	939.5	0.41	3,500	955.7	2.22	3,600	936.4	2,330	938.5	0.26	3,094	956.1	2.12	3,600
200	100	wd.	lrg.	3	1,314.5	490	1,274.7	−2.97	3,012	1,278.1	−2.68	3,059	1,326.1	358	1,282.7	−3.24	2,683	1,283.8	−3.21	3,209
200	100	wd.	lrg.	10	1,522.5	466	1,419.6	−6.42	3,218	1,437.4	−5.43	3,600	1,495.3	411	1,396.2	−6.35	3,258	1,438.4	−3.52	3,600
200	100	wd.	lrg.	100	1,696.9	289	1,553.2	−8.46	3,105	1,583.0	−6.66	3,600	1,678.9	274	1,533.1	−8.62	3,083	1,607.8	−4.10	3,600
400	20	nrr.	sm.	3	489.5	68	481.8	−1.51	400	481.8	−1.51	309	489.9	53	482.0	−1.70	352	484.0	−1.18	309
400	20	nrr.	sm.	10	493.5	47	478.2	−3.10	725	480.2	−2.69	733	499.7	47	476.9	−4.66	722	481.4	−3.74	747
400	20	nrr.	sm.	100	530.6	34	504.4	−5.00	819	513.8	−3.09	900	547.3	21	506.4	−7.58	837	509.5	−7.01	900
400	20	wd.	sm.	3	454.7	362	446.5	−1.64	486	445.6	−1.87	395	456.7	316	446.3	−2.13	516	445.6	−2.30	384
400	20	wd.	sm.	10	470.2	387	449.9	−4.15	900	453.1	−3.59	900	480.9	311	455.3	−5.35	900	452.5	−5.84	900
400	20	wd.	sm.	100	502.4	307	470.3	−6.31	900	477.7	−4.84	900	513.3	178	471.7	−7.98	900	480.3	−6.31	900
400	60	nrr.	sm.	3	886.1	253	886.0	0.00	257	886.0	0.00	226	886.3	165	886.0	−0.03	182	886.0	−0.03	271
400	60	nrr.	sm.	10	953.8	234	953.7	−0.01	258	953.8	−0.01	876	954.9	160	953.9	−0.10	158	953.7	−0.13	1,200
400	60	nrr.	sm.	100	941.2	439	938.6	−0.33	874	939.0	−0.27	1,994	943.9	294	938.6	−0.63	792	938.9	−0.59	2,202
400	60	nrr.	lrg.	3	2,703.3	10	2,424.3	−10.26	411	2,432.4	−9.90	352	2,505.7	11	2,318.9	−7.39	609	2,306.3	−7.91	690

Table C.4: Comparison of hybrid ALNS algorithms with different packing heuristics; extended problem variants (3L-VRPTW, RL/SL, C1); average TTDs, deviations from benchmark and computing times; separated by instance classes, loading approaches and variants of the hybrid algorithm (*continued*)

					ALNS														
					rear loading							**side loading**							
instance parameters					DBLF		LS_DBLF			LS_OS			DBLF		LS_DBLF			LS_OS	
					∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct	∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct
m	n	TW	it.	typ	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]
400	60	nrr.	lrg.	10	2,458.0	29	1,841.8	−24.87	1,684	1,844.5	−24.62	1,996	2,560.8	16	1,889.9	−26.26	981	1,903.9	−25.71	2,128
400	60	wd.	sm.	3	739.3	2,844	740.6	0.20	3,027	739.7	0.07	3,118	744.8	2,759	742.6	−0.24	2,984	738.4	−0.87	2,987
400	60	wd.	sm.	10	677.7	3,588	680.4	0.54	3,600	695.7	2.80	3,600	685.9	3,594	680.2	−0.68	3,600	697.3	1.73	3,600
400	60	wd.	sm.	100	773.4	3,600	771.8	−0.19	3,600	795.2	2.76	3,600	780.2	3,475	775.1	−0.66	3,600	794.3	1.81	3,600
400	60	wd.	lrg.	3	2,402.7	16	2,129.9	−10.94	447	2,127.7	−10.96	449	2,136.0	19	1,985.5	−6.69	796	2,000.1	−6.22	651
400	60	wd.	lrg.	10	2,423.6	20	1,855.5	−23.40	2,270	1,853.3	−23.49	2,869	2,682.7	28	1,970.7	−26.54	1,187	1,924.8	−28.25	2,470
400	60	wd.	lrg.	100	3,004.2	15	2,254.2	−24.98	744	2,242.6	−25.36	1,567								
400	100	nrr.	sm.	3	1,341.6	328	1,341.6	0.00	316	1,341.6	0.00	418	1,341.6	208	1,341.6	0.00	216	1,341.6	0.00	533
400	100	nrr.	sm.	10	1,277.7	454	1,277.0	−0.04	453	1,278.1	0.03	1,400	1,278.0	270	1,277.0	−0.07	296	1,277.2	−0.07	1,742
400	100	nrr.	sm.	100	1,272.6	481	1,272.1	−0.04	512	1,272.4	−0.01	2,230	1,272.8	260	1,271.8	−0.09	289	1,273.0	0.01	2,593
400	100	nrr.	lrg.	3	2,414.6	183	2,174.9	−8.95	2,139	2,185.5	−8.25	2,072	2,334.7	187	2,167.3	−7.23	2,304	2,174.3	−6.81	2,336
400	100	nrr.	lrg.	10	2,885.6	125	2,397.8	−16.90	2,088	2,400.7	−16.74	2,811	2,821.0	145	2,366.9	−16.19	2,237	2,380.2	−15.65	3,299
400	100	nrr.	lrg.	100	2,965.9	125	2,396.7	−19.14	2,806	2,393.6	−19.23	3,376	2,962.0	146	2,390.1	−19.31	2,999	2,430.4	−17.93	3,420
400	100	wd.	sm.	3	902.1	3,600	917.8	1.76	3,600	916.0	1.54	3,600	903.6	3,600	917.7	1.53	3,600	909.7	0.68	3,600
400	100	wd.	sm.	10	1,036.2	3,600	1,058.8	2.24	3,600	1,085.5	4.80	3,600	1,035.3	3,600	1,050.8	1.55	3,600	1,076.7	4.03	3,600
400	100	wd.	sm.	100	930.4	3,600	965.1	3.80	3,600	993.4	6.74	3,600	938.8	3,600	957.0	1.93	3,600	986.5	5.02	3,600
400	100	wd.	lrg.	3	2,422.1	338	2,219.7	−8.09	2,745	2,220.0	−7.88	3,273	2,326.4	308	2,192.2	−5.27	2,439	2,234.0	−3.45	2,991
400	100	wd.	lrg.	10	2,701.8	293	2,370.3	−11.84	2,639	2,423.9	−9.63	3,217	2,528.3	320	2,201.6	−12.03	3,331	2,299.0	−8.10	3,600
400	100	wd.	lrg.	100	2,881.7	228	2,370.8	−17.65	3,600	2,479.5	−13.77	3,600	2,877.9	214	2,362.8	−17.85	3,600	2,489.1	−13.39	3,600
total					1,230.0	798	1,136.5	−4.63	1,512	1,141.9	−4.18	1,816	1,160.6	720	1,086.2	−4.18	1,459	1,094.1	−3.63	1,910

Table C.5: Comparison of hybrid ALNS algorithms with different packing heuristics; extended problem variants (3L-VRPCB(TW), RL/SL, C1); average TTDs, deviations from benchmark and computing times; separated by instance classes, loading approaches and variants of the hybrid algorithm

						ALNS														
						rear loading							**side loading**							
instance parameters						DBLF		LS_DBLF			LS_OS			DBLF		LS_DBLF			LS_OS	
						∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct	∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct
m	n	%LH	TW	it.	typ	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]
200	20	50	-	sm.	3	382.6	35	382.6	0.00	131	382.6	0.00	153	382.6	22	382.6	0.00	132	382.6	0.00	117
200	20	50	-	sm.	10	407.1	36	405.9	−0.30	400	405.7	−0.36	631	407.9	28	405.6	−0.57	460	405.9	−0.50	741
200	20	50	-	sm.	100	400.1	19	399.0	−0.27	722	400.6	0.10	837	400.9	14	399.0	−0.47	684	399.5	−0.35	832
200	20	50	nrr.	sm.	3	514.6	1	514.6	0.00	2	514.6	0.00	2	514.6	1	514.6	0.00	3	514.6	0.00	1
200	20	50	nrr.	sm.	10	545.1	1	545.1	0.00	1	545.1	0.00	2	545.1	1	545.1	0.00	1	545.1	0.00	3
200	20	50	nrr.	sm.	100	496.8	1	496.8	0.00	2	496.8	0.00	5	496.9	1	496.8	−0.01	1	496.8	−0.01	7
200	20	50	wd.	sm.	3	470.2	4	470.2	0.00	4	470.2	0.00	3	470.2	3	470.2	0.00	3	470.2	0.00	4
200	20	50	wd.	sm.	10	483.3	9	483.3	0.00	133	483.3	0.00	402	483.3	6	483.3	0.00	132	483.3	0.00	387
200	20	50	wd.	sm.	100	479.4	9	479.4	−0.01	63	479.4	−0.01	437	481.2	5	479.4	−0.38	72	479.4	−0.38	383
200	20	80	-	sm.	3	374.8	211	374.1	−0.19	519	373.3	−0.41	376	375.7	187	373.8	−0.54	486	374.1	−0.45	444
200	20	80	-	sm.	10	381.6	170	379.8	−0.48	836	382.5	0.23	879	382.5	162	379.0	−0.94	858	381.6	−0.26	900
200	20	80	-	sm.	100	387.4	276	373.9	−3.54	900	375.6	−3.07	900	391.7	163	374.0	−4.49	900	376.7	−3.83	900
200	20	80	nrr.	sm.	3	475.8	3	475.8	0.00	3	475.8	0.00	3	475.8	2	475.8	0.00	2	475.8	0.00	4
200	20	80	nrr.	sm.	10	509.5	3	509.5	0.00	3	509.5	0.00	8	509.5	2	509.5	0.00	2	509.5	0.00	10
200	20	80	nrr.	sm.	100	503.5	4	500.3	−0.71	12	500.3	−0.71	66	506.4	4	500.3	−1.35	27	500.3	−1.35	71
200	20	80	wd.	sm.	3	418.4	67	417.8	−0.14	404	417.8	−0.12	375	417.8	59	417.8	−0.01	394	417.8	−0.01	411
200	20	80	wd.	sm.	10	401.9	164	395.2	−1.66	900	400.2	−0.45	900	403.6	140	395.3	−2.05	900	398.1	−1.42	900
200	20	80	wd.	sm.	100	444.3	168	438.9	−1.37	778	438.9	−1.37	900	448.6	40	438.9	−2.43	784	438.9	−2.43	900
200	60	50	-	sm.	3	674.9	1,709	673.4	−0.22	2,549	674.0	−0.13	2,457	673.0	995	676.1	0.46	2,266	676.5	0.52	2,505
200	60	50	-	sm.	10	683.8	1,357	683.8	0.01	3,203	688.7	0.72	3,407	683.6	927	683.5	−0.01	3,174	689.8	0.92	3,526

Table C.5: Comparison of hybrid ALNS algorithms with different packing heuristics; extended problem variants (3L-VRPCB(TW), RL/SL, C1); average TTDs, deviations from benchmark and computing times; separated by instance classes, loading approaches and variants of the hybrid algorithm (*continued*)

						ALNS													
						rear loading							**side loading**						
instance parameters						DBLF		LS_DBLF			LS_OS			DBLF		LS_DBLF			LS_OS	
						∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct	∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct
m	n	%LH	TW	it.	typ	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]
200	60	50	-	sm.	100	669.8	1,101	670.0	0.03	3,526	679.0	1.35	3,600	670.8	718	669.0	−0.26	3,474	676.1	0.76	3,600
200	60	50	-	lrg.	3	975.4	35	901.0	−7.42	893	907.6	−6.71	710	990.7	39	904.2	−8.33	707	902.7	−8.53	880
200	60	50	-	lrg.	10	1,122.8	47	930.4	−16.84	982	928.3	−17.00	1,988	1,113.3	41	925.4	−16.60	973	918.5	−17.21	2,292
200	60	50	-	lrg.	100	1,157.0	39	943.4	−18.43	1,095	929.5	−19.61	2,554	1,161.7	42	941.7	−18.78	1,041	920.8	−20.61	3,003
200	60	50	nrr.	sm.	3	1,045.7	11	1,045.7	0.00	11	1,045.7	0.00	18	1,045.7	10	1,045.7	0.00	10	1,045.7	0.00	17
200	60	50	nrr.	sm.	10	1,119.0	14	1,119.0	0.00	14	1,119.0	0.00	35	1,119.0	10	1,119.0	0.00	10	1,119.0	0.00	40
200	60	50	nrr.	sm.	100	1,193.4	12	1,193.5	0.01	12	1,193.4	0.00	36	1,193.4	10	1,193.4	0.00	9	1,193.4	0.00	38
200	60	50	nrr.	lrg.	3	1,406.8	9	1,327.0	−5.29	27	1,339.5	−4.42	34	1,415.8	9	1,327.8	−5.94	23	1,326.6	−6.00	38
200	60	50	nrr.	lrg.	10	1,301.6	12	1,104.5	−14.72	89	1,102.7	−14.71	206	1,293.6	13	1,111.6	−13.69	86	1,097.3	−14.69	231
200	60	50	nrr.	lrg.	100	1,299.5	12	1,112.0	−14.55	128	1,106.8	−14.97	459	1,340.6	14	1,112.5	−17.07	118	1,101.6	−17.90	684
200	60	50	wd.	sm.	3	903.9	291	903.9	0.00	805	904.2	0.04	876	903.9	153	904.2	0.04	741	903.9	0.00	859
200	60	50	wd.	sm.	10	848.0	430	848.8	0.11	1,068	848.0	0.00	1,864	848.0	284	847.8	−0.02	955	848.6	0.08	1,892
200	60	50	wd.	sm.	100	865.3	328	865.8	0.08	1,122	867.5	0.34	2,169	864.9	194	866.7	0.20	820	865.1	0.03	2,512
200	60	50	wd.	lrg.	3	1,052.7	23	990.0	−5.46	357	996.9	−4.73	303	1,063.4	25	992.5	−6.41	351	994.6	−6.24	395
200	60	50	wd.	lrg.	10	1,222.7	31	1,033.1	−15.25	801	1,032.7	−15.25	1,358	1,209.4	32	1,029.5	−14.50	683	1,026.4	−14.79	1,763
200	60	50	wd.	lrg.	100	1,270.3	23	1,062.4	−16.45	520	1,053.4	−17.15	1,313	1,265.6	23	1,070.9	−15.30	482	1,049.4	−17.05	1,987
200	60	80	-	sm.	3	626.7	2,795	630.5	0.59	3,051	627.3	0.10	2,813	626.6	2,451	632.0	0.85	3,191	632.6	0.94	2,839
200	60	80	-	sm.	10	619.0	2,749	627.8	1.43	3,553	636.0	2.74	3,600	622.6	2,064	628.9	1.01	3,540	632.2	1.56	3,567
200	60	80	-	sm.	100	660.9	2,494	676.1	2.29	3,600	683.2	3.36	3,600	663.6	1,526	672.5	1.32	3,600	685.3	3.26	3,600
200	60	80	-	lrg.	3	1,269.5	78	1,142.5	−9.59	1,616	1,129.2	−10.59	1,933	1,202.7	74	1,137.5	−5.43	1,499	1,141.5	−5.10	2,216

Table C.5: Comparison of hybrid ALNS algorithms with different packing heuristics; extended problem variants (3L-VRPCB(TW), RL/SL, C1); average TTDs, deviations from benchmark and computing times; separated by instance classes, loading approaches and variants of the hybrid algorithm (*continued*)

						ALNS													
						rear loading							**side loading**						
instance parameters						DBLF	LS_DBLF			LS_OS			DBLF	LS_DBLF			LS_OS		
						∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct	∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct
m	n	%LH	TW	it.	typ	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]
200	60	80	-	lrg.	10	1,204.0	93	1,052.9	−12.39	1,922	1,053.1	−12.29	2,806	1,196.1	80	1,044.4	−12.49	1,926	1,054.7	−11.56	3,062
200	60	80	-	lrg.	100	1,274.8	82	1,084.9	−14.88	1,985	1,073.5	−15.78	3,525	1,286.7	74	1,080.7	−15.99	1,834	1,080.2	−16.03	3,569
200	60	80	nrr.	sm.	3	1,102.2	20	1,102.2	0.00	20	1,102.2	0.00	29	1,102.2	14	1,102.2	0.00	14	1,102.2	0.00	27
200	60	80	nrr.	sm.	10	1,133.3	19	1,133.3	0.00	22	1,133.3	0.00	51	1,133.3	14	1,133.3	0.00	13	1,133.3	0.00	52
200	60	80	nrr.	sm.	100	1,025.1	30	1,025.1	0.00	31	1,025.1	0.00	119	1,025.1	17	1,024.0	−0.13	18	1,025.1	0.00	157
200	60	80	nrr.	lrg.	3	1,508.0	18	1,354.0	−9.87	140	1,345.4	−10.38	204	1,437.4	16	1,346.5	−6.27	131	1,355.7	−5.64	347
200	60	80	nrr.	lrg.	10	1,446.2	21	1,247.3	−13.38	267	1,240.4	−13.85	556	1,298.8	23	1,164.5	−10.28	388	1,155.9	−10.95	1,038
200	60	80	nrr.	lrg.	100	1,478.5	23	1,247.8	−15.43	352	1,217.9	−17.44	840	1,489.3	23	1,226.0	−17.63	321	1,225.7	−17.66	1,149
200	60	80	wd.	sm.	3	718.2	1,042	718.3	0.01	1,045	718.1	−0.02	774	718.4	798	718.1	−0.06	776	718.1	−0.06	1,244
200	60	80	wd.	sm.	10	739.3	1,252	738.6	−0.11	2,549	742.2	0.44	3,528	739.7	714	738.6	−0.16	2,129	740.7	0.16	3,537
200	60	80	wd.	sm.	100	836.5	973	835.9	−0.12	1,860	837.7	0.08	3,414	835.8	303	835.1	−0.12	1,438	837.9	0.24	3,559
200	60	80	wd.	lrg.	3	1,262.7	60	1,124.8	−10.48	1,410	1,114.5	−11.40	1,867	1,184.9	58	1,134.8	−4.17	1,281	1,136.2	−3.71	1,857
200	60	80	wd.	lrg.	10	1,245.7	57	1,096.2	−11.74	1,159	1,089.8	−12.31	1,914	1,254.9	52	1,086.0	−13.31	1,032	1,085.4	−13.33	2,219
200	60	80	wd.	lrg.	100	1,354.6	44	1,138.2	−15.90	1,008	1,126.0	−16.86	2,379	1,366.0	39	1,142.5	−16.43	930	1,123.2	−17.82	2,645
200	100	50	-	sm.	3	868.6	3,247	876.0	0.86	3,279	874.5	0.68	3,246	869.1	2,895	875.7	0.76	3,290	878.2	1.04	3,285
200	100	50	-	sm.	10	869.1	3,045	875.7	0.76	3,593	892.8	2.73	3,600	870.4	2,654	878.0	0.88	3,600	887.1	1.92	3,600
200	100	50	-	sm.	100	873.0	2,653	891.0	2.07	3,600	918.4	5.24	3,600	874.2	2,036	885.8	1.32	3,600	920.2	5.26	3,600
200	100	50	-	lrg.	3	1,094.4	211	1,030.1	−5.84	2,755	1,035.4	−5.29	2,876	1,107.5	191	1,051.9	−4.80	2,213	1,038.5	−5.91	3,123
200	100	50	-	lrg.	10	1,212.6	251	1,086.6	−10.29	3,025	1,086.9	−10.27	3,512	1,200.1	206	1,084.2	−9.45	2,839	1,086.2	−9.29	3,600
200	100	50	-	lrg.	100	1,299.0	201	1,147.8	−11.60	2,687	1,143.6	−11.91	3,600	1,286.5	191	1,150.7	−10.51	2,510	1,137.9	−11.51	3,600

Table C.5: Comparison of hybrid ALNS algorithms with different packing heuristics; extended problem variants (3L-VRPCB(TW), RL/SL, C1); average TTDs, deviations from benchmark and computing times; separated by instance classes, loading approaches and variants of the hybrid algorithm (*continued*)

						ALNS													
						rear loading							**side loading**						
instance parameters						DBLF		LS_DBLF			LS_OS			DBLF		LS_DBLF			LS_OS	
						∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct	∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct
m	n	%LH	TW	it.	typ	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]
200	100	50	nrr.	sm.	3	1,612.4	40	1,612.4	0.00	38	1,612.4	0.00	56	1,612.4	35	1,612.4	0.00	34	1,612.4	0.00	55
200	100	50	nrr.	sm.	10	1,448.4	42	1,448.5	0.00	41	1,447.9	−0.03	95	1,447.9	35	1,447.9	0.00	34	1,447.9	0.00	100
200	100	50	nrr.	sm.	100	1,481.5	45	1,481.5	0.00	44	1,481.5	0.00	108	1,481.5	37	1,481.5	0.00	35	1,481.5	0.00	113
200	100	50	nrr.	lrg.	3	1,597.0	45	1,557.1	−2.61	110	1,554.7	−2.80	170	1,612.7	44	1,568.2	−2.90	108	1,560.1	−3.42	281
200	100	50	nrr.	lrg.	10	1,709.1	43	1,575.7	−7.76	149	1,582.5	−7.37	431	1,699.3	45	1,579.3	−7.13	138	1,570.5	−7.64	547
200	100	50	nrr.	lrg.	100	1,808.5	40	1,682.6	−7.05	127	1,679.2	−7.24	526	1,835.6	46	1,695.9	−7.71	120	1,675.3	−8.81	745
200	100	50	wd.	sm.	3	1,092.6	417	1,092.5	−0.01	398	1,092.6	0.00	443	1,093.6	287	1,093.6	0.00	289	1,093.2	−0.03	539
200	100	50	wd.	sm.	10	1,209.3	451	1,210.1	0.07	421	1,210.1	0.07	1,868	1,210.0	258	1,210.1	0.01	259	1,210.1	0.01	2,410
200	100	50	wd.	sm.	100	1,247.0	519	1,249.5	0.19	619	1,252.0	0.45	2,301	1,249.1	261	1,246.3	−0.20	331	1,249.1	0.00	2,930
200	100	50	wd.	lrg.	3	1,200.0	128	1,137.2	−5.24	1,797	1,147.9	−4.35	2,200	1,210.4	117	1,162.5	−3.92	1,421	1,143.8	−5.35	2,530
200	100	50	wd.	lrg.	10	1,348.5	149	1,254.9	−6.93	1,800	1,248.2	−7.34	3,101	1,337.3	121	1,247.1	−6.53	1,661	1,246.8	−6.49	3,323
200	100	50	wd.	lrg.	100	1,506.0	109	1,370.6	−9.07	1,037	1,351.5	−10.32	3,143	1,502.7	101	1,363.8	−9.18	987	1,350.0	−10.10	3,420
200	100	80	-	sm.	3	842.8	3,414	857.5	1.75	3,429	855.5	1.52	3,416	842.2	3,334	855.9	1.64	3,368	855.0	1.54	3,422
200	100	80	-	sm.	10	824.6	3,600	846.3	2.64	3,600	860.5	4.33	3,600	824.1	3,600	849.0	3.00	3,600	852.1	3.38	3,600
200	100	80	-	sm.	100	830.3	3,597	868.3	4.58	3,600	872.0	5.02	3,600	830.1	3,483	866.1	4.34	3,600	875.8	5.50	3,600
200	100	80	-	lrg.	3	1,236.4	338	1,209.8	−2.19	2,076	1,214.1	−1.89	2,556	1,259.3	302	1,216.6	−3.51	2,125	1,216.2	−3.69	2,819
200	100	80	-	lrg.	10	1,415.2	337	1,302.4	−7.89	3,076	1,320.5	−6.59	3,600	1,379.7	316	1,291.1	−6.34	2,863	1,321.3	−4.15	3,600
200	100	80	-	lrg.	100	1,449.7	370	1,332.0	−8.10	3,447	1,357.8	−6.30	3,600	1,430.3	352	1,322.8	−7.50	3,432	1,382.3	−3.33	3,600
200	100	80	nrr.	sm.	3	1,398.7	51	1,398.7	0.00	50	1,398.7	0.00	94	1,398.7	39	1,398.7	0.00	39	1,398.7	0.00	93
200	100	80	nrr.	sm.	10	1,401.9	59	1,401.9	0.00	56	1,402.0	0.00	152	1,402.0	41	1,402.0	0.00	41	1,402.0	0.00	151

Table C.5: Comparison of hybrid ALNS algorithms with different packing heuristics; extended problem variants (3L-VRPCB(TW), RL/SL, C1); average TTDs, deviations from benchmark and computing times; separated by instance classes, loading approaches and variants of the hybrid algorithm (*continued*)

						ALNS													
						rear loading							**side loading**						
instance parameters						DBLF		LS_DBLF			LS_OS			DBLF		LS_DBLF			LS_OS	
						∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct	∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct
m	n	%LH	TW	it.	typ	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]
200	100	80	nrr.	sm.	100	1,556.8	47	1,556.4	−0.02	50	1,558.1	0.08	164	1,558.1	36	1,558.5	0.02	37	1,558.1	0.00	163
200	100	80	nrr.	lrg.	3	1,645.6	58	1,605.3	−2.55	165	1,607.2	−2.38	457	1,658.4	55	1,617.4	−2.35	174	1,600.6	−3.42	582
200	100	80	nrr.	lrg.	10	1,620.3	83	1,493.8	−7.76	413	1,488.3	−8.08	2,459	1,589.5	63	1,491.6	−6.13	384	1,484.6	−6.58	2,476
200	100	80	nrr.	lrg.	100	1,779.4	68	1,636.3	−7.98	313	1,619.7	−8.95	1,544	1,747.8	58	1,622.8	−7.11	311	1,617.4	−7.42	1,830
200	100	80	wd.	sm.	3	1,132.0	994	1,132.0	0.00	1,094	1,132.0	0.00	1,126	1,132.0	677	1,132.1	0.01	685	1,132.0	0.00	1,136
200	100	80	wd.	sm.	10	1,000.1	1,845	999.3	−0.06	1,859	1,000.8	0.09	3,305	1,000.6	1,135	998.9	−0.13	1,300	1,001.8	0.13	3,541
200	100	80	wd.	sm.	100	1,100.0	1,850	1,100.7	0.07	2,734	1,106.5	0.66	3,600	1,100.2	1,086	1,101.2	0.09	2,442	1,109.2	0.88	3,599
200	100	80	wd.	lrg.	3	1,302.2	184	1,270.7	−2.50	1,514	1,269.3	−2.65	1,941	1,318.0	153	1,276.6	−3.16	1,288	1,275.1	−3.29	2,252
200	100	80	wd.	lrg.	10	1,556.9	197	1,435.8	−7.73	1,867	1,436.3	−7.69	3,561	1,519.9	182	1,425.4	−6.16	1,936	1,412.2	−6.99	3,600
200	100	80	wd.	lrg.	100	1,624.1	187	1,503.6	−7.44	1,901	1,510.4	−6.95	3,383	1,611.3	169	1,495.0	−7.13	1,883	1,503.5	−6.48	3,600
400	20	50	-	sm.	3	414.0	99	411.0	−0.65	492	409.9	−0.91	423	416.2	84	410.2	−1.31	461	412.0	−0.89	410
400	20	50	-	sm.	10	401.2	65	392.7	−2.02	812	396.2	−1.16	814	401.5	49	392.7	−2.15	813	396.0	−1.39	814
400	20	50	-	sm.	100	431.9	79	416.6	−3.47	900	423.9	−1.79	900	437.5	41	416.4	−4.71	900	419.3	−4.07	900
400	20	50	nrr.	sm.	3	466.9	9	465.5	−0.33	369	465.5	−0.33	363	467.6	7	465.5	−0.47	364	465.5	−0.47	363
400	20	50	nrr.	sm.	10	609.1	1	607.2	−0.35	304	607.2	−0.35	361	609.6	1	607.2	−0.41	316	607.2	−0.41	362
400	20	50	nrr.	sm.	100	562.9	2	562.7	−0.05	260	562.7	−0.05	399	580.6	1	562.7	−3.15	245	562.7	−3.15	401
400	20	50	wd.	sm.	3	525.6	34	521.6	−0.59	379	521.4	−0.63	366	526.1	25	521.4	−0.70	379	521.4	−0.70	365
400	20	50	wd.	sm.	10	487.9	22	481.5	−1.46	765	481.5	−1.46	900	493.0	19	481.5	−2.48	809	481.5	−2.48	817
400	20	50	wd.	sm.	100	470.4	26	454.0	−3.46	900	455.3	−3.15	900	479.9	16	454.3	−5.41	900	455.9	−5.08	900
400	20	80	-	sm.	3	430.2	269	426.5	−0.85	485	426.7	−0.85	458	432.8	235	428.6	−0.87	561	427.1	−1.25	468

Table C.5: Comparison of hybrid ALNS algorithms with different packing heuristics; extended problem variants (3L-VRPCB(TW), RL/SL, C1); average TTDs, deviations from benchmark and computing times; separated by instance classes, loading approaches and variants of the hybrid algorithm (*continued*)

						ALNS														
						rear loading							**side loading**							
instance parameters						DBLF		LS_DBLF			LS_OS			DBLF		LS_DBLF			LS_OS	
						∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct	∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct
m	n	%LH	TW	it.	typ	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]
400	20	80	-	sm.	10	418.6	430	406.5	−2.80	900	410.7	−1.91	900	422.8	281	407.1	−3.64	900	410.4	−2.93	900
400	20	80	-	sm.	100	455.8	327	427.0	−6.25	900	444.8	−2.39	900	470.3	198	432.0	−8.14	900	438.9	−6.66	900
400	20	80	nrr.	sm.	3	623.7	4	618.6	−0.78	210	616.1	−1.14	187	623.7	3	618.6	−0.83	264	616.1	−1.19	192
400	20	80	nrr.	sm.	10	478.6	32	468.6	−2.11	557	470.0	−1.85	758	486.0	23	467.0	−3.90	715	471.0	−3.03	839
400	20	80	nrr.	sm.	100	568.8	25	541.0	−5.38	897	556.2	−2.21	900	581.9	16	544.7	−6.96	870	551.3	−5.64	900
400	20	80	wd.	sm.	3	419.2	225	418.3	−0.24	382	417.0	−0.57	369	422.4	131	417.0	−1.31	407	417.0	−1.31	373
400	20	80	wd.	sm.	10	467.6	229	451.6	−3.40	900	457.9	−1.93	900	474.2	152	451.7	−4.73	900	453.2	−4.35	900
400	20	80	wd.	sm.	100	500.4	122	477.5	−4.63	900	486.5	−2.71	900	508.1	77	477.2	−5.99	900	483.7	−4.64	900
400	60	50	-	sm.	3	698.2	2,299	699.1	0.16	2,687	702.1	0.58	2,497	698.5	2,148	702.2	0.56	2,712	697.8	−0.09	2,466
400	60	50	-	sm.	10	688.1	2,891	692.2	0.62	3,600	701.7	2.02	3,600	689.8	2,451	693.5	0.59	3,600	700.4	1.60	3,600
400	60	50	-	sm.	100	686.0	2,244	688.5	0.34	3,600	701.4	2.27	3,600	689.7	1,354	684.5	−0.75	3,600	701.6	1.71	3,600
400	60	50	-	lrg.	3	1,740.4	17	1,508.4	−13.28	609	1,526.1	−12.47	448	1,642.4	21	1,478.3	−10.30	733	1,470.3	−10.77	743
400	60	50	-	lrg.	10	1,781.5	24	1,321.7	−25.71	1,360	1,313.1	−26.19	2,269	1,746.5	31	1,272.8	−27.18	1,769	1,258.8	−27.93	3,049
400	60	50	-	lrg.	100	2,022.7	17	1,595.3	−21.12	906	1,558.1	−22.96	1,516								
400	60	50	nrr.	sm.	3	1,087.5	54	1,087.5	0.00	55	1,087.5	0.00	54	1,087.5	37	1,087.5	0.00	40	1,087.5	0.00	55
400	60	50	nrr.	sm.	10	1,202.8	34	1,202.8	0.00	30	1,202.8	0.00	111	1,202.8	22	1,202.8	0.00	22	1,202.8	0.00	133
400	60	50	nrr.	sm.	100	1,146.6	50	1,146.0	−0.05	54	1,146.0	−0.05	309	1,147.1	35	1,146.6	−0.06	37	1,146.0	−0.11	421
400	60	50	nrr.	lrg.	3	1,870.0	11	1,694.7	−9.33	185	1,701.4	−8.95	145	1,868.2	8	1,743.0	−6.72	166	1,732.3	−7.30	197
400	60	50	nrr.	lrg.	10	2,002.6	12	1,486.1	−25.44	303	1,484.6	−25.51	744	1,926.2	17	1,483.4	−22.99	718	1,452.6	−24.59	1,294
400	60	50	nrr.	lrg.	100	2,137.0	11	1,642.8	−23.12	278	1,596.4	−25.30	505								

Table C.5: Comparison of hybrid ALNS algorithms with different packing heuristics; extended problem variants (3L-VRPCB(TW), RL/SL, C1); average TTDs, deviations from benchmark and computing times; separated by instance classes, loading approaches and variants of the hybrid algorithm (*continued*)

						ALNS														
						rear loading							**side loading**							
instance parameters						DBLF		LS_DBLF			LS_OS			DBLF		LS_DBLF			LS_OS	
						∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct	∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct
m	n	%LH	TW	it.	typ	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]
400	60	50	wd.	sm.	3	920.7	1,054	919.2	−0.17	1,844	919.4	−0.15	1,788	920.1	683	919.2	−0.10	1,692	920.0	0.00	1,767
400	60	50	wd.	sm.	10	776.4	1,473	782.6	0.85	3,514	790.9	1.97	3,600	777.4	1,014	775.6	−0.22	3,102	791.0	1.86	3,600
400	60	50	wd.	sm.	100	938.6	994	932.8	−0.60	3,600	949.0	1.10	3,600	942.4	758	931.9	−1.14	3,600	957.5	1.76	3,600
400	60	50	wd.	lrg.	3	1,650.3	14	1,468.3	−11.40	451	1,466.0	−11.57	333	1,694.3	15	1,536.4	−9.71	303	1,543.6	−9.38	430
400	60	50	wd.	lrg.	10	1,865.9	15	1,485.2	−20.81	785	1,477.1	−21.25	1,319	1,723.4	15	1,278.5	−25.82	1,382	1,259.6	−26.91	3,600
400	60	50	wd.	lrg.	100	2,030.8	21	1,677.9	−17.38	747	1,632.4	−19.62	916								
400	60	80	-	sm.	3	657.7	3,164	660.4	0.47	3,271	660.5	0.47	3,167	664.2	3,122	664.8	0.19	3,451	660.3	−0.49	3,096
400	60	80	-	sm.	10	670.0	3,579	684.9	2.28	3,600	685.3	2.29	3,600	672.2	3,514	681.3	1.38	3,600	698.0	3.91	3,600
400	60	80	-	sm.	100	682.0	3,599	685.5	0.52	3,600	707.4	3.76	3,600	688.4	3,466	691.8	0.48	3,600	700.4	1.74	3,600
400	60	80	-	lrg.	3	2,127.3	17	1,941.8	−8.31	350	1,927.4	−8.79	307	1,961.1	16	1,859.5	−4.68	371	1,867.6	−4.44	594
400	60	80	-	lrg.	10	2,042.5	83	1,546.3	−24.12	3,529	1,569.1	−22.92	3,600	2,016.2	28	1,595.8	−20.37	1,984	1,574.0	−21.33	3,600
400	60	80	-	lrg.	100	2,407.7	11	1,771.8	−26.30	508	1,775.6	−26.14	989								
400	60	80	nrr.	sm.	3	975.6	109	975.6	0.00	112	975.6	0.00	143	975.6	74	975.6	0.00	77	975.6	0.00	191
400	60	80	nrr.	sm.	10	1,021.2	123	1,021.2	0.00	130	1,021.2	0.00	496	1,021.2	82	1,021.2	0.00	80	1,021.2	0.00	740
400	60	80	nrr.	sm.	100	1,001.6	197	1,000.7	−0.11	263	1,000.6	−0.12	1,560	1,002.2	110	1,000.7	−0.17	262	1,000.6	−0.18	1,752
400	60	80	nrr.	lrg.	3	2,277.6	12	2,096.8	−8.11	236	2,086.9	−8.42	285	2,126.6	12	2,004.0	−6.09	393	1,986.6	−6.95	666
400	60	80	nrr.	lrg.	10	1,918.0	40	1,525.1	−20.48	3,175	1,586.7	−17.27	3,600	2,179.7	9	1,762.5	−19.14	315	1,761.9	−19.17	958
400	60	80	wd.	sm.	3	818.2	2,347	821.0	0.42	2,669	820.5	0.40	2,855	819.3	1,939	821.5	0.30	2,881	820.3	0.16	2,722
400	60	80	wd.	sm.	10	834.3	2,993	829.9	−0.46	3,600	841.1	0.83	3,600	833.8	2,216	828.9	−0.55	3,600	836.8	0.35	3,600
400	60	80	wd.	sm.	100	822.3	2,888	819.4	−0.36	3,600	841.4	2.37	3,600	826.7	2,264	823.8	−0.35	3,600	854.7	3.53	3,600

Table C.5: Comparison of hybrid ALNS algorithms with different packing heuristics; extended problem variants (3L-VRPCB(TW), RL/SL, C1); average TTDs, deviations from benchmark and computing times; separated by instance classes, loading approaches and variants of the hybrid algorithm (*continued*)

						ALNS													
						rear loading							**side loading**						
instance parameters						DBLF		LS_DBLF			LS_OS			DBLF		LS_DBLF			LS_OS	
						∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct	∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct
m	n	%LH	TW	it.	typ	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]
400	60	80	wd.	lrg.	3	2,095.6	16	1,883.5	−9.21	180	1,878.5	−9.29	166	1,561.5	15	1,539.3	−1.44	30	1,539.3	−1.44	172
400	60	80	wd.	lrg.	10	2,094.9	51	1,522.0	−27.35	3,488	1,553.7	−25.84	3,600	2,279.3	18	1,765.0	−22.57	769	1,736.8	−23.80	1,612
400	60	80	wd.	lrg.	100	2,523.1	14	1,824.7	−27.68	383	1,801.0	−28.62	1,307								
400	100	50	-	sm.	3	878.2	3,582	900.3	2.51	3,590	899.5	2.42	3,600	874.9	3,495	906.6	3.62	3,446	900.0	2.87	3,588
400	100	50	-	sm.	10	877.8	3,600	914.4	4.17	3,600	914.6	4.18	3,600	881.1	3,476	909.6	3.24	3,600	922.5	4.71	3,600
400	100	50	-	sm.	100	897.5	3,597	927.7	3.38	3,600	945.2	5.31	3,600	896.4	3,514	927.5	3.49	3,600	948.9	5.87	3,600
400	100	50	-	lrg.	3	1,831.8	181	1,597.0	−12.42	2,445	1,586.1	−12.95	2,825	1,754.1	189	1,593.3	−9.31	2,672	1,597.8	−9.08	2,991
400	100	50	-	lrg.	10	1,985.4	152	1,638.3	−17.31	2,683	1,634.4	−17.43	3,476	1,985.4	171	1,569.1	−20.76	2,799	1,577.4	−20.24	3,595
400	100	50	-	lrg.	100	2,020.7	153	1,595.7	−20.90	3,409	1,589.7	−21.17	3,600	2,089.7	155	1,599.8	−23.30	3,410	1,596.9	−23.44	3,600
400	100	50	nrr.	sm.	3	1,583.3	104	1,583.3	0.00	105	1,583.3	0.00	135	1,583.3	69	1,583.3	0.00	72	1,583.3	0.00	127
400	100	50	nrr.	sm.	10	1,499.0	132	1,499.0	0.00	142	1,498.0	−0.06	324	1,497.8	92	1,499.2	0.10	91	1,498.0	0.02	400
400	100	50	nrr.	sm.	100	1,447.0	176	1,447.0	0.00	176	1,447.0	0.00	647	1,448.4	115	1,447.7	−0.04	106	1,447.7	−0.04	984
400	100	50	nrr.	lrg.	3	2,077.8	56	1,826.4	−11.27	815	1,819.5	−11.60	932	1,981.2	57	1,833.6	−7.43	823	1,821.7	−8.06	1,131
400	100	50	nrr.	lrg.	10	2,261.2	68	1,899.9	−15.86	925	1,883.2	−16.61	1,343	2,266.4	61	1,884.7	−16.89	881	1,875.8	−17.38	1,840
400	100	50	nrr.	lrg.	100	2,350.2	65	1,948.3	−17.01	692	1,919.9	−18.29	1,723	2,460.3	89	1,971.2	−20.11	681	1,938.3	−21.33	1,211
400	100	50	wd.	sm.	3	1,122.9	2,997	1,130.1	0.70	3,024	1,128.7	0.56	2,760	1,123.5	2,306	1,129.1	0.52	2,933	1,133.1	0.93	2,847
400	100	50	wd.	sm.	10	1,346.3	2,136	1,351.2	0.36	2,989	1,357.8	0.85	3,463	1,348.3	1,569	1,351.1	0.21	2,930	1,368.0	1.48	3,554
400	100	50	wd.	sm.	100	1,207.1	3,391	1,219.3	1.03	3,600	1,247.4	3.36	3,600	1,208.0	2,603	1,218.1	0.80	3,600	1,257.2	4.17	3,600
400	100	50	wd.	lrg.	3	1,933.4	125	1,743.4	−10.03	2,268	1,735.0	−10.37	2,725	1,895.5	120	1,737.2	−8.33	2,220	1,745.6	−7.96	2,923
400	100	50	wd.	lrg.	10	2,037.9	118	1,703.8	−16.37	2,360	1,699.4	−16.49	3,179	1,959.4	182	1,575.1	−19.21	2,618	1,593.2	−18.20	3,326

Table C.5: Comparison of hybrid ALNS algorithms with different packing heuristics; extended problem variants (3L-VRPCB(TW), RL/SL, C1); average TTDs, deviations from benchmark and computing times; separated by instance classes, loading approaches and variants of the hybrid algorithm (*continued*)

instance parameters						**ALNS**													
						rear loading							**side loading**						
						DBLF		LS_DBLF			LS_OS			DBLF		LS_DBLF			LS_OS	
m	n	%LH	TW	it.	typ	∅TTD* [DU]	∅ct [s]	∅TTD [DU]	∅dev [%]	∅ct [s]	∅TTD [DU]	∅dev [%]	∅ct [s]	∅TTD* [DU]	∅ct [s]	∅TTD [DU]	∅dev [%]	∅ct [s]	∅TTD [DU]	∅dev [%]	∅ct [s]
400	100	50	wd.	lrg.	100	2,151.3	117	1,744.9	−18.80	2,334	1,720.8	−19.89	3,575	2,273.1	141	1,743.4	−23.29	2,766	1,723.8	−24.13	3,600
400	100	80	-	sm.	3	856.1	3,600	882.3	3.09	3,600	877.2	2.49	3,600	853.7	3,600	881.7	3.32	3,600	873.7	2.39	3,600
400	100	80	-	sm.	10	856.3	3,600	887.0	3.62	3,600	901.2	5.26	3,600	856.4	3,600	885.1	3.40	3,600	898.3	4.92	3,600
400	100	80	-	sm.	100	877.9	3,600	905.0	3.11	3,600	943.7	7.46	3,600	884.0	3,600	917.9	3.83	3,600	936.6	5.92	3,600
400	100	80	-	lrg.	3	1,824.8	370	1,698.6	−6.54	2,678	1,721.1	−5.13	2,887	1,781.2	315	1,680.6	−5.61	2,303	1,701.3	−4.25	2,788
400	100	80	-	lrg.	10	2,192.8	343	1,908.4	−12.81	2,814	1,953.6	−10.68	3,259	2,276.2	320	1,923.0	−15.07	3,031	1,973.7	−12.57	3,595
400	100	80	-	lrg.	100	2,456.1	268	2,035.9	−16.93	3,593	2,102.1	−14.24	3,600	2,427.2	307	2,011.9	−17.03	3,600	2,129.2	−12.16	3,600
400	100	80	nrr.	sm.	3	1,548.0	175	1,548.0	0.00	212	1,548.0	0.00	218	1,548.0	131	1,548.0	0.00	120	1,548.0	0.00	297
400	100	80	nrr.	sm.	10	1,512.8	238	1,513.1	0.02	226	1,512.8	0.00	655	1,512.3	116	1,512.3	0.00	151	1,512.8	0.03	865
400	100	80	nrr.	sm.	100	1,566.4	338	1,565.9	−0.04	275	1,565.3	−0.08	1,277	1,566.4	142	1,565.9	−0.03	196	1,566.0	−0.01	1,493
400	100	80	nrr.	lrg.	3	1,981.7	200	1,880.8	−5.16	1,562	1,896.0	−4.23	1,362	2,016.0	161	1,881.6	−6.73	1,463	1,878.9	−6.69	1,413
400	100	80	nrr.	lrg.	10	2,402.2	111	1,974.1	−17.65	1,393	1,971.5	−17.74	2,089	2,542.0	99	2,173.4	−14.46	1,123	2,155.3	−15.20	1,518
400	100	80	nrr.	lrg.	100	2,560.0	119	2,113.6	−17.35	2,272	2,088.1	−18.36	3,414	2,589.3	162	2,074.9	−19.82	2,993	2,083.7	−19.46	3,600
400	100	80	wd.	sm.	3	1,115.5	3,490	1,135.3	1.67	3,600	1,132.8	1.45	3,600	1,121.0	3,402	1,135.6	1.23	3,600	1,127.7	0.57	3,600
400	100	80	wd.	sm.	10	1,194.8	3,598	1,207.7	1.17	3,600	1,214.2	1.71	3,600	1,196.2	3,463	1,207.9	1.09	3,600	1,227.9	2.78	3,600
400	100	80	wd.	sm.	100	1,091.0	3,570	1,114.4	2.18	3,600	1,150.9	5.66	3,600	1,093.0	3,393	1,118.6	2.46	3,600	1,168.0	6.81	3,600
400	100	80	wd.	lrg.	3	1,986.6	255	1,767.4	−10.55	2,796	1,780.9	−9.72	3,293	1,894.0	228	1,736.5	−8.13	2,378	1,759.4	−6.90	2,970
400	100	80	wd.	lrg.	10	2,335.9	223	2,011.6	−13.73	2,281	2,060.5	−11.39	2,710	2,129.2	346	1,839.8	−12.74	2,845	1,937.9	−8.08	3,600
400	100	80	wd.	lrg.	100	2,517.2	202	2,073.7	−17.48	3,390	2,101.4	−16.35	3,600	2,400.0	186	1,975.3	−17.66	3,540	2,075.1	−13.50	3,600
total						1,104.1	886	1,029.3	−4.31	1,680	1,032.6	−3.89	1,948	1,047.0	803	988.7	−3.69	1,652	992.7	−3.27	2,031

Table C.6: Comparison of hybrid ALNS algorithms with different packing heuristics; extended problem variants (3L-VRPMB(TW), LSP/SL, C1); average TTDs, deviations from benchmark and computing times; separated by instance classes, loading approaches and variants of the hybrid algorithm

					ALNS														
					loading space partition								**side loading**						
instance parameters					**DBLF**		**LS_DBLF**			**LS_OS**			**DBLF^{SL}**		**LS_DBLF^{SL}**			**LS_OS**		
					∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct	∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct
n	%LH	TW	it.	typ	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]
20	50	-	sm.	3	362.3	40	351.5	−2.88	653	354.4	−2.33	560	322.2	391	320.5	−0.46	557	314.6	−2.21	351
20	50	-	sm.	10	376.6	28	366.1	−2.88	879	370.6	−1.83	891	351.3	537	347.4	−1.03	772	341.7	−2.67	851
20	50	-	sm.	100	369.2	23	357.6	−3.17	743	366.8	−0.78	848	361.8	769	352.0	−2.59	900	345.9	−4.38	900
20	50	nrr.	sm.	3	531.2	2	495.4	−6.18	49	497.7	−5.71	47	438.8	9	438.3	−0.09	9	435.1	−0.83	4
20	50	nrr.	sm.	10	487.0	2	477.9	−1.85	1	477.9	−1.85	1	465.9	6	465.9	0.00	6	465.3	−0.19	7
20	50	nrr.	sm.	100	472.6	3	466.9	−1.19	4	476.5	1.06	8	440.5	23	440.5	0.00	24	437.4	−0.77	67
20	50	wd.	sm.	3	406.2	10	399.1	−1.74	328	399.9	−1.55	363	387.3	132	387.3	0.00	225	387.3	−0.01	163
20	50	wd.	sm.	10	412.6	9	392.6	−4.86	581	399.1	−3.33	756	402.1	291	400.0	−0.55	521	393.9	−1.98	576
20	50	wd.	sm.	100	436.7	7	417.2	−4.22	490	425.1	−2.46	578	430.5	224	427.9	−0.57	721	422.2	−1.98	893
20	80	-	sm.	3	404.8	86	391.5	−3.02	900	393.7	−2.54	900	342.7	527	341.4	−0.38	670	340.3	−0.71	624
20	80	-	sm.	10	391.2	79	373.2	−4.60	900	377.7	−3.34	900	343.6	813	334.4	−2.63	900	337.3	−1.71	900
20	80	-	sm.	100	404.9	62	378.2	−6.55	900	389.3	−3.95	900	369.6	721	347.9	−5.74	900	361.2	−2.29	900
20	80	nrr.	sm.	3	460.7	3	453.5	−1.62	339	448.3	−2.65	302	440.7	11	440.7	0.00	10	439.1	−0.39	5
20	80	nrr.	sm.	10	480.4	4	475.0	−1.46	210	477.3	−0.84	355	473.6	9	473.6	0.00	10	473.6	0.00	73
20	80	nrr.	sm.	100	477.3	6	464.9	−2.79	555	463.1	−3.08	699	461.5	37	461.2	−0.08	83	460.4	−0.36	312
20	80	wd.	sm.	3	439.1	36	417.9	−4.60	875	422.9	−3.50	887	375.6	279	373.2	−0.63	586	370.3	−1.44	501
20	80	wd.	sm.	10	401.0	54	383.0	−4.41	900	383.9	−4.17	900	353.1	881	342.0	−3.06	900	344.9	−2.22	900
20	80	wd.	sm.	100	414.4	25	398.4	−3.96	900	399.7	−3.64	900	404.6	437	388.8	−4.03	900	392.7	−2.89	900
60	50	-	sm.	3	538.3	980	544.3	1.16	2,203	547.1	1.67	2,072	519.4	2,929	519.1	−0.06	3,006	511.0	−1.58	2,427
60	50	-	sm.	10	549.5	779	555.2	1.06	2,605	561.8	2.32	2,891	542.8	2,796	544.0	0.22	3,269	531.6	−2.06	3,596

Table C.6: Comparison of hybrid ALNS algorithms with different packing heuristics; extended problem variants (3L-VRPMB(TW), LSP/SL, C1); average TTDs, deviations from benchmark and computing times; separated by instance classes, loading approaches and variants of the hybrid algorithm (*continued*)

					ALNS														
					loading space partition								**side loading**						
instance parameters					**DBLF**		**LS_DBLF**			**LS_OS**			**DBLF^{SL}**		**LS_DBLF^{SL}**			**LS_OS**		
					∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct	∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct
n	%LH	TW	it.	typ	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]
60	50	-	sm.	100	545.3	837	541.2	−0.75	1,989	545.9	0.08	2,851	548.3	2,980	548.3	0.00	3,471	541.3	−1.29	3,600
60	50	-	lrg.	3	1,279.8	52	1,112.8	−13.39	466	1,116.5	−13.20	1,441	980.9	242	890.3	−8.43	1,952	893.1	−8.41	1,642
60	50	-	lrg.	10	1,508.5	55	1,145.5	−24.07	486	1,097.3	−27.26	1,139	1,135.3	238	972.2	−13.94	2,021	969.1	−14.17	3,293
60	50	-	lrg.	100									1,154.2	222	995.0	−13.73	1,957	979.5	−15.02	3,544
60	50	nrr.	sm.	3	975.7	24	904.0	−7.15	15	904.3	−7.12	18	892.2	24	892.2	0.00	26	892.2	0.00	40
60	50	nrr.	sm.	10	846.9	31	804.9	−4.89	17	803.2	−5.08	32	929.1	34	929.1	0.00	38	929.1	0.00	74
60	50	nrr.	sm.	100	1,179.8	23	1,045.2	−11.15	14	1,043.2	−11.32	18	997.6	27	997.6	0.00	30	998.0	0.04	65
60	50	nrr.	lrg.	3	1,450.2	22	1,314.5	−9.99	60	1,287.2	−11.82	473	1,346.5	22	1,231.5	−7.25	57	1,229.8	−7.48	71
60	50	nrr.	lrg.	10									1,251.7	53	1,100.8	−11.47	304	1,099.7	−11.62	730
60	50	nrr.	lrg.	100	1,751.6	30	1,625.4	−7.21	114	1,561.5	−10.85	1,912	1,283.2	55	1,123.7	−12.46	362	1,108.4	−13.69	976
60	50	wd.	sm.	3	731.0	226	722.8	−1.13	448	725.4	−0.75	456	719.6	1,012	719.6	0.00	956	716.4	−0.51	724
60	50	wd.	sm.	10	697.3	423	700.2	0.53	876	699.8	0.51	1,055	687.8	1,610	688.4	0.10	1,649	682.9	−0.86	2,331
60	50	wd.	sm.	100	696.1	334	695.3	−0.05	674	691.7	−0.53	1,019	687.4	1,350	688.2	0.09	1,805	683.6	−0.56	3,428
60	50	wd.	lrg.	3	1,245.9	50	1,020.0	−18.13	477	1,005.4	−19.30	561	1,023.0	159	953.9	−6.35	1,352	965.3	−5.19	1,236
60	50	wd.	lrg.	10	1,347.7	49	1,159.9	−13.94	381	1,097.4	−18.58	1,286	1,235.8	182	1,065.8	−13.57	1,558	1,049.1	−14.86	3,245
60	50	wd.	lrg.	100									1,261.4	152	1,088.6	−13.63	1,150	1,064.0	−15.61	2,703
60	80	-	sm.	3	550.0	2,193	545.8	−0.71	3,600	547.9	−0.28	3,586	518.6	3,111	517.8	−0.16	3,400	515.3	−0.60	2,877
60	80	-	sm.	10	544.2	1,702	538.6	−1.04	3,600	540.3	−0.73	3,600	522.1	3,371	523.1	0.19	3,600	523.2	0.29	3,600
60	80	-	sm.	100	574.4	1,397	562.3	−2.10	3,600	562.4	−2.08	3,600	552.1	3,561	552.4	0.05	3,600	559.6	1.36	3,600

Table C.6: Comparison of hybrid ALNS algorithms with different packing heuristics; extended problem variants (3L-VRPMB(TW), LSP/SL, C1); average TTDs, deviations from benchmark and computing times; separated by instance classes, loading approaches and variants of the hybrid algorithm (*continued*)

					ALNS														
					loading space partition								**side loading**						
instance parameters					**DBLF**		**LS_DBLF**			**LS_OS**			**DBLF^{SL}**		**LS_DBLF^{SL}**			**LS_OS**		
					∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct	∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct
n	%LH	TW	it.	typ	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]
60	80	-	lrg.	3	2,164.1	35	2,048.4	−5.49	283	2,046.3	−5.54	2,686	1,319.8	178	1,149.0	−12.04	1,847	1,138.0	−12.79	2,094
60	80	-	lrg.	10	1,717.8	50	1,503.7	−12.38	744	1,440.3	−16.12	2,114	1,211.3	239	1,073.4	−11.03	2,092	1,069.0	−11.36	2,807
60	80	-	lrg.	100	2,257.4	52	1,797.4	−20.38	408	1,726.7	−23.51	3,600	1,302.5	198	1,119.0	−14.03	2,179	1,095.1	−15.87	3,501
60	80	nrr.	sm.	3	951.3	25	946.5	−0.47	24	946.5	−0.47	29	946.5	36	946.5	0.00	37	946.5	0.00	37
60	80	nrr.	sm.	10	1,014.5	19	994.1	−1.93	15	994.1	−1.93	31	994.1	26	994.1	0.00	29	994.1	0.00	53
60	80	nrr.	sm.	100	903.8	26	902.1	−0.17	23	902.5	−0.12	57	903.7	47	902.9	−0.10	51	902.1	−0.20	120
60	80	nrr.	lrg.	3	2,094.0	21	1,949.6	−7.25	66	1,948.4	−7.31	897	1,513.5	42	1,317.5	−12.11	220	1,315.9	−12.04	249
60	80	nrr.	lrg.	10	2,021.3	29	1,715.1	−15.15	242	1,676.5	−17.06	1,817	1,426.0	58	1,236.1	−13.14	344	1,224.2	−13.94	656
60	80	nrr.	lrg.	100									1,460.6	74	1,252.5	−14.28	472	1,235.7	−15.42	1,059
60	80	wd.	sm.	3	607.6	1,044	608.1	0.12	2,740	605.0	−0.42	2,476	595.4	2,467	594.9	−0.10	2,752	594.9	−0.09	1,597
60	80	wd.	sm.	10	626.0	652	617.6	−1.48	3,184	622.7	−0.68	3,517	618.4	2,578	619.4	0.17	3,068	615.0	−0.60	3,272
60	80	wd.	sm.	100	690.7	664	685.8	−0.84	3,398	688.5	−0.37	3,600	693.5	2,113	693.6	0.00	2,875	687.3	−0.97	3,598
60	80	wd.	lrg.	3	2,135.2	33	2,017.5	−5.74	237	2,012.3	−5.97	2,779	1,290.2	167	1,130.3	−11.80	1,791	1,124.5	−12.33	1,841
60	80	wd.	lrg.	10	1,490.5	53	1,318.9	−11.51	733	1,240.2	−16.79	1,302	1,236.2	196	1,095.5	−11.11	1,606	1,088.5	−11.58	2,119
60	80	wd.	lrg.	100	2,348.7	41	1,830.7	−22.06	253	1,735.4	−26.11	2,140	1,351.2	137	1,163.2	−13.91	1,354	1,150.6	−14.88	2,464
100	50	-	sm.	3	680.5	2,788	693.4	1.92	2,548	689.3	1.35	2,218	657.6	3,574	661.8	0.65	3,560	648.5	−1.37	3,219
100	50	-	sm.	10	698.5	3,072	716.0	2.50	2,677	730.2	4.53	3,410	678.4	3,600	686.1	1.13	3,600	680.4	0.25	3,600
100	50	-	sm.	100	698.9	2,561	710.9	1.77	2,622	722.2	3.36	3,449	691.7	3,600	695.4	0.55	3,600	696.8	0.73	3,600
100	50	-	lrg.	3	1,624.7	165	1,420.7	−12.64	1,076	1,382.9	−14.89	3,532	1,075.8	921	1,030.4	−3.94	3,295	1,042.1	−2.78	3,329
100	50	-	lrg.	10	1,788.1	161	1,492.8	−16.47	878	1,431.2	−19.99	3,600	1,181.4	872	1,122.0	−4.87	3,430	1,111.6	−5.86	3,600

Table C.6: Comparison of hybrid ALNS algorithms with different packing heuristics; extended problem variants (3L-VRPMB(TW), LSP/SL, C1); average TTDs, deviations from benchmark and computing times; separated by instance classes, loading approaches and variants of the hybrid algorithm (*continued*)

					ALNS														
					loading space partition							**side loading**							
instance parameters					**DBLF**		**LS_DBLF**			**LS_OS**			**DBLF^{SL}**		**LS_DBLF^{SL}**			**LS_OS**		
					∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct	∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct
n	%LH	TW	it.	typ	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]
100	50	-	lrg.	100	1,866.3	163	1,569.3	−15.81	809	1,517.8	−18.52	3,600	1,278.2	709	1,194.6	−6.44	3,451	1,209.7	−5.26	3,600
100	50	nrr.	sm.	3	1,526.3	65	1,345.5	−11.44	43	1,345.5	−11.44	50	1,320.7	67	1,320.7	0.00	75	1,320.7	0.00	135
100	50	nrr.	sm.	10	1,381.5	66	1,213.8	−10.89	48	1,213.6	−10.91	70	1,194.9	84	1,194.4	−0.04	88	1,195.1	0.02	188
100	50	nrr.	sm.	100	1,385.6	68	1,264.2	−8.32	50	1,264.7	−8.29	68	1,234.5	88	1,232.8	−0.13	99	1,234.5	0.01	212
100	50	nrr.	lrg.	3	1,817.6	71	1,580.5	−13.00	182	1,555.0	−14.22	1,434	1,485.2	89	1,435.6	−3.36	244	1,431.7	−3.57	694
100	50	nrr.	lrg.	10	2,127.9	70	1,658.8	−21.76	185	1,576.2	−25.66	1,912	1,589.0	92	1,512.7	−4.75	278	1,491.9	−6.13	1,100
100	50	nrr.	lrg.	100	2,017.5	65	1,695.9	−15.85	73	1,641.8	−18.54	804	1,650.4	77	1,582.2	−4.24	249	1,567.2	−5.14	1,310
100	50	wd.	sm.	3	904.6	521	879.2	−2.73	945	883.0	−2.34	969	848.1	1,982	846.9	−0.12	1,837	845.4	−0.31	1,686
100	50	wd.	sm.	10	1,066.0	442	1,000.8	−5.79	189	1,001.8	−5.71	385	958.1	2,285	957.2	−0.08	2,371	957.0	−0.13	3,587
100	50	wd.	sm.	100	1,036.0	415	1,012.1	−2.18	214	1,020.8	−1.30	476	971.1	2,122	970.5	−0.05	2,051	968.9	−0.25	3,574
100	50	wd.	lrg.	3	1,615.9	164	1,426.8	−11.80	700	1,394.8	−13.62	3,346	1,146.9	691	1,104.6	−3.63	3,289	1,112.2	−2.94	3,236
100	50	wd.	lrg.	10	1,903.2	134	1,491.3	−21.68	705	1,425.7	−25.24	3,548	1,284.7	687	1,217.4	−5.04	2,676	1,192.7	−6.84	3,451
100	50	wd.	lrg.	100	2,032.3	133	1,654.2	−18.47	526	1,588.0	−21.68	3,600	1,471.2	400	1,378.8	−6.19	2,083	1,353.5	−7.89	3,600
100	80	-	sm.	3	672.3	3,288	682.7	1.57	3,554	676.8	0.68	3,572	658.8	3,579	667.7	1.35	3,600	655.0	−0.56	3,485
100	80	-	sm.	10	681.3	3,386	680.2	−0.15	3,600	693.8	1.84	3,600	673.7	3,600	685.7	1.78	3,600	684.2	1.55	3,600
100	80	-	sm.	100	696.9	2,674	698.6	0.23	3,600	711.7	2.14	3,600	692.5	3,600	703.0	1.49	3,600	699.7	1.02	3,600
100	80	-	lrg.	3	1,811.9	157	1,739.9	−4.27	1,644	1,756.5	−3.29	3,019	1,239.6	622	1,200.1	−3.03	2,674	1,197.9	−3.19	2,832
100	80	-	lrg.	10	2,135.5	154	1,953.5	−8.48	1,373	1,962.1	−8.05	3,600	1,406.5	579	1,323.2	−5.75	3,260	1,340.8	−4.61	3,600
100	80	-	lrg.	100	2,184.9	155	1,953.8	−10.39	1,418	1,982.0	−9.11	3,600	1,454.3	594	1,363.0	−6.24	3,575	1,383.5	−4.84	3,600

Table C.6: Comparison of hybrid ALNS algorithms with different packing heuristics; extended problem variants (3L-VRPMB(TW), LSP/SL, C1); average TTDs, deviations from benchmark and computing times; separated by instance classes, loading approaches and variants of the hybrid algorithm (*continued*)

					ALNS													
					loading space partition								**side loading**					
instance parameters					**DBLF**		**LS_DBLF**			**LS_OS**			**DBLF^{SL}**		**LS_DBLF^{SL}**			**LS_OS**		
					∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct	∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct
n	%LH	TW	it.	typ	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]
100	80	nrr.	sm.	3	1,210.4	80	1,211.0	0.11	59	1,210.5	0.05	103	1,207.9	103	1,207.9	0.00	108	1,207.9	0.00	142
100	80	nrr.	sm.	10	1,238.2	80	1,233.6	−0.34	59	1,233.5	−0.36	129	1,232.8	111	1,231.9	−0.08	111	1,231.5	−0.12	198
100	80	nrr.	sm.	100	1,346.9	63	1,342.7	−0.29	49	1,342.3	−0.33	110	1,342.3	76	1,342.7	0.04	85	1,342.2	0.00	184
100	80	nrr.	lrg.	3	1,940.4	83	1,847.8	−4.67	372	1,834.0	−5.23	3,459	1,545.5	112	1,506.7	−2.62	335	1,503.2	−2.61	756
100	80	nrr.	lrg.	10	2,159.6	80	1,935.7	−10.20	412	1,918.2	−11.06	3,096	1,596.2	123	1,480.6	−7.04	551	1,461.5	−8.20	2,329
100	80	nrr.	lrg.	100	2,324.5	82	2,030.8	−12.40	337	1,957.7	−15.60	3,047	1,700.7	108	1,592.8	−6.33	455	1,571.7	−7.55	1,701
100	80	wd.	sm.	3	914.3	1,363	917.1	0.39	1,322	915.2	0.13	1,290	912.2	2,363	912.6	0.06	2,496	912.6	0.06	2,086
100	80	wd.	sm.	10	804.4	2,255	801.5	−0.34	2,633	808.0	0.48	3,527	802.1	3,473	801.3	−0.09	3,518	801.2	−0.12	3,600
100	80	wd.	sm.	100	907.7	1,552	905.9	−0.20	2,692	914.3	0.81	3,363	906.9	3,119	908.9	0.24	3,291	911.2	0.48	3,600
100	80	wd.	lrg.	3	1,825.0	130	1,746.7	−4.60	1,196	1,745.3	−4.75	2,500	1,283.0	458	1,229.8	−3.88	2,744	1,208.3	−5.62	2,414
100	80	wd.	lrg.	10	2,333.4	125	2,078.0	−11.13	890	2,068.9	−11.56	3,600	1,543.7	396	1,447.9	−6.14	2,299	1,432.8	−7.10	3,596
100	80	wd.	lrg.	100	2,372.9	129	2,015.7	−14.80	1,010	2,029.2	−14.23	3,600	1,611.9	371	1,515.2	−5.98	2,285	1,509.1	−6.23	3,600
total					1,002.3	686	931.9	−4.49	1,331	928.3	−4.35	1,971	911.9	1,115	866.3	−3.56	1,818	862.6	−3.97	2,080

Table C.7: Comparison of hybrid ALNS algorithms with different packing heuristics; extended problem variants (3L-VRPSDP(TW), LSP/SL, C1); average TTDs, deviations from benchmark and computing times; separated by instance classes, loading approaches and variants of the hybrid algorithm

					ALNS														
					loading space partition								**side loading**						
instance parameters					**DBLF**		**LS_DBLF**			**LS_OS**			**DBLF^{SL}**		**LS_DBLF^{SL}**			**LS_OS**		
					∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct	∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct
n	%LH	TW	it.	typ	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]
20	50	-	sm.	3	348.6	207	338.6	−2.63	900	338.8	−2.65	853	329.1	534	329.1	0.00	562	319.1	−2.68	359
20	50	-	sm.	10	368.6	112	353.9	−3.83	884	357.1	−2.97	900	373.7	612	373.7	0.00	650	346.2	−7.15	893
20	50	-	sm.	100	360.6	72	346.8	−3.73	900	349.1	−3.20	900	380.6	719	379.3	−0.32	793	362.8	−4.64	900
20	50	nrr.	sm.	3	450.2	2	443.0	−1.62	43	442.1	−1.81	40	439.7	11	439.7	0.00	10	432.3	−1.70	2
20	50	nrr.	sm.	10	462.8	2	462.8	0.00	2	462.8	0.00	4	465.1	7	465.1	0.00	7	462.8	−0.53	5
20	50	nrr.	sm.	100	436.7	4	436.6	−0.01	5	436.6	−0.01	14	445.8	29	445.8	0.00	30	437.0	−2.11	51
20	50	wd.	sm.	3	394.6	30	392.1	−0.65	453	392.4	−0.60	395	388.3	138	388.3	0.00	152	386.6	−0.44	77
20	50	wd.	sm.	10	394.8	39	389.6	−1.32	900	390.3	−1.14	900	406.5	291	403.4	−0.75	311	390.6	−3.77	590
20	50	wd.	sm.	100	416.5	24	413.2	−0.77	271	413.2	−0.77	820	428.8	380	428.8	0.00	363	416.8	−2.74	427
20	80	-	sm.	3	403.0	113	393.7	−2.12	900	392.8	−2.29	900	364.8	739	364.4	−0.12	749	345.2	−5.36	582
20	80	-	sm.	10	387.3	94	375.3	−2.97	900	377.7	−2.35	900	373.9	789	370.8	−0.81	842	349.3	−6.68	900
20	80	-	sm.	100	404.9	71	386.5	−4.57	900	392.8	−3.06	900	396.9	866	383.0	−3.38	900	370.6	−6.51	900
20	80	nrr.	sm.	3	450.3	4	443.5	−1.55	336	443.5	−1.55	310	446.1	13	444.0	−0.52	16	439.1	−1.66	5
20	80	nrr.	sm.	10	481.1	6	475.9	−1.27	200	475.9	−1.27	233	479.1	42	478.8	−0.07	40	473.6	−1.49	96
20	80	nrr.	sm.	100	464.6	7	458.9	−1.51	337	458.9	−1.51	548	467.9	94	462.4	−1.37	160	458.7	−2.40	147
20	80	wd.	sm.	3	432.5	46	418.1	−3.02	894	420.2	−2.60	883	389.5	424	386.9	−0.69	418	370.1	−4.80	547
20	80	wd.	sm.	10	395.9	72	380.4	−3.90	900	383.4	−3.13	900	384.8	645	373.0	−3.07	874	362.1	−5.96	900
20	80	wd.	sm.	100	416.7	23	402.1	−3.50	900	406.1	−2.52	900	416.7	490	404.6	−2.98	643	386.8	−7.18	900
60	50	-	sm.	3	537.7	1,242	532.5	−0.96	3,528	532.2	−1.01	3,453	523.7	2,530	523.7	−0.01	2,725	512.5	−2.08	2,537
60	50	-	sm.	10	547.6	1,012	545.0	−0.49	3,097	545.8	−0.34	3,496	554.5	3,027	554.2	−0.05	2,936	536.7	−3.15	3,338

Table C.7: Comparison of hybrid ALNS algorithms with different packing heuristics; extended problem variants (3L-VRPSDP(TW), LSP/SL, C1); average TTDs, deviations from benchmark and computing times; separated by instance classes, loading approaches and variants of the hybrid algorithm (*continued*)

					ALNS														
					loading space partition								**side loading**						
instance parameters					**DBLF**		**LS_DBLF**			**LS_OS**			**DBLF^{SL}**		**LS_DBLF^{SL}**			**LS_OS**		
					∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct	∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct
n	%LH	TW	it.	typ	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]
60	50	-	sm.	100	541.0	803	538.6	−0.44	3,221	538.9	−0.40	3,600	560.4	2,986	559.0	−0.26	3,111	540.5	−3.51	3,593
60	50	-	lrg.	3	1,462.8	46	1,402.4	−4.24	596	1,395.4	−4.46	2,554	1,112.5	318	1,092.1	−1.82	860	1,036.9	−7.31	1,800
60	50	-	lrg.	10	1,543.1	37	1,423.0	−7.73	569	1,397.1	−9.47	3,536	1,392.3	242	1,337.7	−3.79	904	1,323.9	−4.94	3,371
60	50	-	lrg.	100	1,540.0	39	1,380.0	−10.38	550	1,342.4	−12.83	3,600	1,373.7	235	1,321.6	−3.76	927	1,286.3	−6.33	3,586
60	50	nrr.	sm.	3	892.2	15	892.2	0.00	15	892.2	0.00	34	892.2	20	892.2	0.00	22	892.2	0.00	36
60	50	nrr.	sm.	10	929.1	18	929.1	0.00	19	928.8	−0.04	54	929.1	28	929.1	0.00	31	928.8	−0.04	65
60	50	nrr.	sm.	100	997.6	15	997.6	0.00	15	998.0	0.05	43	998.0	22	998.0	0.00	23	998.0	0.00	52
60	50	nrr.	lrg.	3	1,583.6	17	1,494.8	−5.81	86	1,481.0	−6.44	1,160	1,243.6	40	1,216.1	−2.05	68	1,159.3	−7.14	160
60	50	nrr.	lrg.	10	1,577.4	24	1,460.7	−7.52	244	1,435.6	−9.14	3,024	1,405.7	96	1,352.7	−3.48	264	1,365.4	−2.88	1,517
60	50	nrr.	lrg.	100	1,576.2	22	1,396.8	−11.35	202	1,347.3	−14.54	2,827	1,422.4	88	1,361.4	−4.29	270	1,361.1	−4.30	1,846
60	50	wd.	sm.	3	718.7	323	718.6	−0.02	1,232	718.7	−0.02	1,232	718.9	688	719.1	0.03	692	716.3	−0.40	710
60	50	wd.	sm.	10	687.5	594	684.0	−0.58	1,534	684.2	−0.54	2,041	693.7	1,450	690.6	−0.49	1,672	684.6	−1.44	2,138
60	50	wd.	sm.	100	682.7	695	680.8	−0.29	2,362	680.9	−0.28	3,157	690.4	1,311	690.1	−0.03	1,519	682.4	−1.26	3,198
60	50	wd.	lrg.	3	1,450.3	39	1,376.1	−4.99	440	1,365.4	−5.49	2,203	1,178.1	242	1,160.2	−1.50	521	1,147.8	−2.33	1,414
60	50	wd.	lrg.	10	1,578.0	36	1,427.8	−9.39	476	1,382.8	−12.27	3,498	1,508.4	189	1,443.8	−4.26	686	1,412.2	−6.45	3,245
60	50	wd.	lrg.	100	1,607.8	32	1,431.9	−10.91	403	1,385.1	−13.84	3,600	1,454.2	176	1,403.3	−3.36	622	1,353.8	−6.89	3,381
60	80	-	sm.	3	548.2	2,066	546.9	−0.11	3,600	543.1	−0.85	3,563	524.3	3,137	523.9	−0.08	3,271	515.9	−1.53	2,613
60	80	-	sm.	10	542.6	1,886	533.8	−1.62	3,600	536.0	−1.20	3,600	536.7	3,191	537.8	0.20	3,498	521.0	−2.87	3,600
60	80	-	sm.	100	571.6	1,526	564.4	−1.28	3,600	569.6	−0.34	3,600	581.3	3,406	584.0	0.47	3,562	563.1	−3.10	3,600

Table C.7: Comparison of hybrid ALNS algorithms with different packing heuristics; extended problem variants (3L-VRPSDP(TW), LSP/SL, C1); average TTDs, deviations from benchmark and computing times; separated by instance classes, loading approaches and variants of the hybrid algorithm (*continued*)

					ALNS														
					loading space partition							**side loading**							
instance parameters					**DBLF**		**LS_DBLF**			**LS_OS**			**DBLF^{SL}**		**LS_DBLF^{SL}**			**LS_OS**		
					∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct	∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct
n	%LH	TW	it.	typ	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]
60	80	-	lrg.	3	2,088.4	27	1,966.3	−5.85	428	1,917.1	−8.10	3,247	1,399.8	207	1,292.0	−7.22	1,814	1,258.0	−9.10	2,337
60	80	-	lrg.	10	1,703.4	40	1,537.0	−9.81	802	1,494.1	−12.28	3,528	1,356.3	277	1,237.0	−8.59	2,031	1,201.6	−11.26	3,208
60	80	-	lrg.	100	1,905.1	30	1,710.9	−10.16	618	1,653.1	−13.21	3,600	1,428.0	226	1,299.6	−9.01	1,824	1,239.5	−13.25	3,537
60	80	nrr.	sm.	3	946.5	18	946.5	0.00	19	946.5	0.00	32	946.5	29	946.5	0.00	30	946.5	0.00	34
60	80	nrr.	sm.	10	994.4	14	994.1	−0.04	16	994.1	−0.04	33	994.1	23	994.1	0.00	24	994.1	0.00	43
60	80	nrr.	sm.	100	902.1	23	902.1	0.00	25	902.1	0.00	73	903.6	50	903.2	−0.05	55	902.1	−0.19	107
60	80	nrr.	lrg.	3	2,072.9	19	1,902.3	−8.58	197	1,845.5	−11.02	2,386	1,561.5	56	1,431.1	−7.36	269	1,402.4	−8.49	444
60	80	nrr.	lrg.	10	1,743.8	23	1,553.0	−10.92	215	1,495.3	−14.23	1,949	1,529.6	63	1,405.8	−8.05	314	1,367.5	−10.83	751
60	80	nrr.	lrg.	100	1,881.5	21	1,707.1	−9.23	356	1,624.3	−13.62	3,482	1,513.6	81	1,385.6	−8.45	473	1,314.6	−13.16	1,425
60	80	wd.	sm.	3	608.1	1,606	604.2	−0.65	3,019	603.7	−0.75	3,068	597.0	1,959	596.7	−0.05	2,100	594.4	−0.44	1,678
60	80	wd.	sm.	10	622.4	748	620.4	−0.34	3,178	620.3	−0.34	3,547	620.0	2,194	619.8	−0.04	2,073	616.7	−0.56	3,453
60	80	wd.	sm.	100	689.6	695	685.2	−0.69	3,272	686.5	−0.51	3,600	698.1	1,973	699.2	0.20	2,284	684.2	−2.12	3,582
60	80	wd.	lrg.	3	2,113.6	23	2,000.3	−5.19	356	1,946.6	−7.75	3,386	1,443.5	175	1,323.3	−8.12	1,395	1,262.8	−11.94	1,973
60	80	wd.	lrg.	10	1,742.6	38	1,568.4	−10.15	633	1,532.4	−12.06	3,016	1,397.4	241	1,284.5	−7.67	1,681	1,228.6	−11.83	2,729
60	80	wd.	lrg.	100	1,940.8	24	1,739.4	−10.36	485	1,689.4	−13.01	3,581	1,499.2	162	1,371.6	−8.59	1,142	1,322.8	−11.95	2,723
100	50	-	sm.	3	670.1	2,764	670.0	−0.01	3,600	671.3	0.18	3,461	666.0	3,530	667.2	0.17	3,580	654.6	−1.70	3,113
100	50	-	sm.	10	678.0	2,884	679.3	0.20	3,600	693.9	2.34	3,600	688.8	3,462	693.2	0.65	3,489	684.7	−0.57	3,600
100	50	-	sm.	100	685.3	3,093	687.8	0.37	3,600	706.0	3.01	3,600	707.4	3,600	709.0	0.22	3,600	697.6	−1.39	3,600
100	50	-	lrg.	3	1,594.8	190	1,547.3	−3.03	1,688	1,556.1	−2.59	3,557	1,332.7	848	1,320.8	−0.84	2,725	1,299.2	−2.34	3,357
100	50	-	lrg.	10	1,677.6	171	1,606.2	−4.24	1,638	1,627.1	−3.16	3,600	1,399.5	872	1,379.7	−1.44	3,028	1,344.1	−4.24	3,600

Table C.7: Comparison of hybrid ALNS algorithms with different packing heuristics; extended problem variants (3L-VRPSDP(TW), LSP/SL, C1); average TTDs, deviations from benchmark and computing times; separated by instance classes, loading approaches and variants of the hybrid algorithm (*continued*)

					ALNS														
					loading space partition								**side loading**						
instance parameters					**DBLF**		**LS_DBLF**			**LS_OS**			**DBLF^{SL}**		**LS_DBLF^{SL}**			**LS_OS**		
					∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct	∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct
n	%LH	TW	it.	typ	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]
100	50	-	lrg.	100	1,765.0	165	1,677.6	−4.94	1,522	1,707.2	−3.28	3,600	1,501.2	642	1,467.3	−2.25	2,837	1,468.1	−2.25	3,600
100	50	nrr.	sm.	3	1,320.5	50	1,320.5	0.00	54	1,320.5	0.00	110	1,320.7	64	1,320.7	0.00	73	1,320.5	−0.02	128
100	50	nrr.	sm.	10	1,194.3	53	1,194.7	0.04	61	1,194.6	0.02	180	1,194.6	86	1,194.0	−0.05	99	1,194.6	0.00	197
100	50	nrr.	sm.	100	1,233.7	58	1,232.8	−0.07	63	1,233.7	0.00	182	1,234.5	89	1,232.7	−0.15	102	1,233.7	−0.07	214
100	50	nrr.	lrg.	3	1,737.0	66	1,641.1	−5.53	311	1,632.5	−6.00	2,136	1,675.4	109	1,661.3	−0.80	204	1,678.3	0.11	741
100	50	nrr.	lrg.	10	1,898.8	62	1,793.7	−5.51	301	1,750.4	−7.77	2,806	1,705.5	122	1,673.0	−1.94	293	1,652.2	−3.31	1,924
100	50	nrr.	lrg.	100	1,927.8	57	1,809.8	−6.13	244	1,747.9	−9.30	2,484	1,758.4	102	1,718.7	−2.28	251	1,667.1	−5.24	1,472
100	50	wd.	sm.	3	851.3	1,215	850.0	−0.15	2,336	848.1	−0.40	2,251	846.8	2,072	848.4	0.16	2,044	845.2	−0.21	1,543
100	50	wd.	sm.	10	956.1	1,122	956.0	−0.02	1,450	956.0	−0.02	3,425	956.6	2,066	954.6	−0.20	2,390	956.4	−0.03	3,583
100	50	wd.	sm.	100	970.3	1,108	968.5	−0.17	2,002	969.5	−0.04	3,489	973.1	2,131	971.6	−0.11	2,288	968.8	−0.43	3,600
100	50	wd.	lrg.	3	1,619.0	133	1,573.9	−2.80	1,182	1,560.2	−3.83	3,591	1,319.1	601	1,301.5	−1.38	2,591	1,239.1	−5.76	3,185
100	50	wd.	lrg.	10	1,621.2	137	1,543.7	−4.78	1,088	1,517.4	−6.49	3,600	1,401.4	654	1,381.9	−1.32	2,099	1,312.4	−6.42	3,513
100	50	wd.	lrg.	100	1,842.5	124	1,725.8	−6.33	996	1,727.5	−6.19	3,600	1,611.9	406	1,580.1	−1.93	1,512	1,558.9	−3.19	3,600
100	80	-	sm.	3	674.6	3,262	678.7	0.61	3,600	678.1	0.52	3,600	661.0	3,600	673.9	1.94	3,600	657.0	−0.58	3,515
100	80	-	sm.	10	682.9	3,266	686.1	0.48	3,600	696.1	1.94	3,600	684.5	3,600	701.0	2.43	3,600	687.7	0.50	3,600
100	80	-	sm.	100	699.9	3,225	703.4	0.52	3,600	708.9	1.33	3,600	702.8	3,600	723.0	2.86	3,600	716.3	1.92	3,600
100	80	-	lrg.	3	1,961.3	178	1,908.3	−2.90	2,000	1,937.5	−1.46	3,295	1,312.5	821	1,282.5	−2.11	2,776	1,264.3	−3.70	3,329
100	80	-	lrg.	10	2,216.8	162	2,078.5	−6.28	1,681	2,128.9	−3.95	3,600	1,521.4	737	1,449.3	−4.67	3,320	1,459.4	−4.24	3,600
100	80	-	lrg.	100	2,159.6	157	2,033.5	−5.88	1,772	2,079.0	−3.79	3,600	1,538.3	715	1,471.0	−4.37	3,573	1,490.0	−3.18	3,600

Table C.7: Comparison of hybrid ALNS algorithms with different packing heuristics; extended problem variants (3L-VRPSDP(TW), LSP/SL, C1); average TTDs, deviations from benchmark and computing times; separated by instance classes, loading approaches and variants of the hybrid algorithm (*continued*)

					ALNS													
					loading space partition								**side loading**					
instance parameters					**DBLF**		**LS_DBLF**			**LS_OS**			**DBLF^{SL}**		**LS_DBLF^{SL}**			**LS_OS**		
					∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct	∅TTD*	∅ct	∅TTD	∅dev	∅ct	∅TTD	∅dev	∅ct
n	%LH	TW	it.	typ	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]	[DU]	[s]	[DU]	[%]	[s]	[DU]	[%]	[s]
100	80	nrr.	sm.	3	1,208.5	55	1,207.5	−0.09	61	1,208.0	−0.05	116	1,208.5	89	1,207.9	−0.06	100	1,208.0	−0.05	126
100	80	nrr.	sm.	10	1,231.9	63	1,232.4	0.05	65	1,232.3	0.04	160	1,231.9	109	1,233.1	0.08	108	1,232.3	0.04	201
100	80	nrr.	sm.	100	1,342.2	55	1,342.7	0.04	58	1,341.3	−0.07	142	1,343.2	83	1,342.3	−0.07	93	1,341.3	−0.15	181
100	80	nrr.	lrg.	3	2,135.8	74	2,060.2	−3.55	476	2,067.1	−3.23	3,600	1,528.7	115	1,498.0	−2.12	372	1,496.1	−2.20	1,119
100	80	nrr.	lrg.	10	2,249.4	73	2,096.5	−6.79	537	2,075.8	−7.71	3,342	1,703.0	158	1,611.4	−5.30	664	1,560.9	−8.37	2,781
100	80	nrr.	lrg.	100	2,246.6	64	2,059.7	−8.28	381	2,015.8	−10.19	3,301	1,757.3	125	1,660.7	−5.47	435	1,614.3	−8.09	2,339
100	80	wd.	sm.	3	912.6	1,659	912.6	0.00	2,159	912.8	0.02	2,206	913.4	2,624	913.8	0.06	2,617	912.6	−0.09	2,174
100	80	wd.	sm.	10	802.3	2,449	801.7	−0.09	3,235	802.8	0.04	3,600	802.4	3,566	801.5	−0.11	3,587	801.6	−0.13	3,600
100	80	wd.	sm.	100	909.4	1,898	907.1	−0.29	3,208	916.8	0.90	3,600	910.7	3,267	913.1	0.30	3,315	913.2	0.23	3,600
100	80	wd.	lrg.	3	1,909.9	127	1,836.6	−4.17	1,379	1,834.8	−4.38	2,625	1,357.0	549	1,311.3	−3.20	2,860	1,269.5	−6.45	2,683
100	80	wd.	lrg.	10	2,320.9	108	2,160.5	−6.98	1,022	2,191.8	−5.68	3,600	1,613.9	453	1,524.0	−5.52	2,620	1,509.2	−6.52	3,600
100	80	wd.	lrg.	100	2,274.8	113	2,134.5	−6.18	1,096	2,160.3	−5.03	3,600	1,697.4	416	1,605.5	−5.46	2,376	1,590.8	−6.23	3,600
total					1,098.1	641	1,051.4	−3.01	1,456	1,048.0	−3.07	2,417	971.1	1,121	946.0	−1.71	1,652	928.9	−3.73	2,151

Table C.8: Comparison of loading approaches; extended problem variants (3L-VRPTW, RL/SL, C1); average TTDs provided by rear loading (benchmark), average deviations of TTDs provided by side loading from benchmarks; separated by instance classes and variants of the hybrid algorithm

					ALNS					
					DBLF		**LS_DBLF**		**LS_OS**	
instance parameters					RL	SL	RL	SL	RL	SL
					∅TTD	∅dev	∅TTD	∅dev	∅TTD	∅dev
m	n	TW	it.	typ	[DU]	[%]	[DU]	[%]	[DU]	[%]
200	20	nrr.	sm.	3	439.84	0.19	438.40	0.05	438.40	0.00
200	20	nrr.	sm.	10	468.17	0.00	468.17	0.00	468.17	0.00
200	20	nrr.	sm.	100	450.08	0.17	447.65	0.00	447.65	0.00
200	20	wd.	sm.	3	391.54	0.02	389.63	0.03	389.80	0.17
200	20	wd.	sm.	10	375.64	0.79	367.65	0.03	368.21	0.04
200	20	wd.	sm.	100	411.23	0.67	397.10	−0.08	400.36	0.17
200	60	nrr.	sm.	3	919.38	0.00	919.38	0.00	919.38	0.00
200	60	nrr.	sm.	10	961.56	0.00	961.56	0.00	961.56	0.00
200	60	nrr.	sm.	100	950.08	0.00	950.27	0.01	950.08	0.00
200	60	nrr.	lrg.	3	1,533.92	−1.78	1,390.91	0.27	1,393.78	−0.04
200	60	nrr.	lrg.	10	1,481.75	−0.83	1,260.71	−0.57	1,255.12	−0.81
200	60	nrr.	lrg.	100	1,546.74	−1.77	1,267.41	−0.10	1,251.15	−0.42
200	60	wd.	sm.	3	656.17	0.13	656.94	−0.09	655.82	0.02
200	60	wd.	sm.	10	650.62	−0.12	648.24	0.20	650.54	0.12
200	60	wd.	sm.	100	682.09	0.35	682.10	0.14	691.03	0.62
200	60	wd.	lrg.	3	1,294.35	−0.73	1,189.30	0.12	1,182.94	0.51
200	60	wd.	lrg.	10	1,426.32	−0.68	1,225.86	−0.25	1,225.77	−0.13
200	60	wd.	lrg.	100	1,543.56	−1.08	1,283.45	−0.38	1,263.53	−0.75
200	100	nrr.	sm.	3	1,265.39	0.00	1,265.39	0.00	1,265.39	0.00
200	100	nrr.	sm.	10	1,214.11	0.00	1,213.87	0.03	1,214.16	0.00
200	100	nrr.	sm.	100	1,288.95	0.02	1,288.83	−0.13	1,288.67	0.00
200	100	nrr.	lrg.	3	1,523.51	2.16	1,470.77	1.01	1,471.51	0.14
200	100	nrr.	lrg.	10	1,675.35	−2.55	1,524.08	−0.51	1,518.31	−0.03
200	100	nrr.	lrg.	100	1,774.94	−0.95	1,592.63	−0.29	1,573.40	−0.62
200	100	wd.	sm.	3	882.86	−0.08	882.14	0.05	881.99	0.01
200	100	wd.	sm.	10	876.50	0.16	876.11	0.16	880.83	−0.01
200	100	wd.	sm.	100	936.22	0.04	939.48	−0.10	955.70	−0.04
200	100	wd.	lrg.	3	1,314.50	1.06	1,274.66	0.74	1,278.07	0.41
200	100	wd.	lrg.	10	1,522.49	−1.79	1,419.60	−1.71	1,437.39	0.19
200	100	wd.	lrg.	100	1,696.87	−1.06	1,553.15	−1.27	1,582.98	1.63
400	20	nrr.	sm.	3	489.47	0.27	481.78	0.06	481.80	0.60
400	20	nrr.	sm.	10	493.46	1.43	478.18	−0.28	480.18	0.26
400	20	nrr.	sm.	100	530.62	3.24	504.42	0.46	513.79	−0.93
400	20	wd.	sm.	3	454.75	0.45	446.46	−0.05	445.63	0.00
400	20	wd.	sm.	10	470.22	2.33	449.87	1.08	453.15	−0.06
400	20	wd.	sm.	100	502.38	2.14	470.32	0.31	477.74	0.57
400	60	nrr.	sm.	3	886.07	0.03	886.03	0.00	886.03	0.00
400	60	nrr.	sm.	10	953.83	0.12	953.74	0.02	953.76	0.00

Table C.8: Comparison of loading approaches; extended problem variants (3L-VRPTW, RL/SL, C1); average TTDs provided by rear loading (benchmark), average deviations of TTDs provided by side loading from benchmarks; separated by instance classes and variants of the hybrid algorithm (*continued*)

					ALNS					
					DBLF		**LS_DBLF**		**LS_OS**	
instance parameters					RL	SL	RL	SL	RL	SL
					∅TTD	∅dev	∅TTD	∅dev	∅TTD	∅dev
m	n	TW	it.	typ	[DU]	[%]	[DU]	[%]	[DU]	[%]
400	60	nrr.	sm.	100	941.16	0.30	938.57	0.00	939.00	−0.02
400	60	nrr.	lrg.	3	2,536.95	−1.25	2,303.42	0.72	2,304.83	0.01
400	60	nrr.	lrg.	10	2,517.90	0.08	1,848.47	−1.64	1,823.20	0.43
400	60	wd.	sm.	3	739.34	0.77	740.61	0.31	739.71	−0.19
400	60	wd.	sm.	10	677.68	1.18	680.44	−0.04	695.68	0.17
400	60	wd.	sm.	100	773.36	0.85	771.84	0.39	795.23	−0.06
400	60	wd.	lrg.	3	1,941.01	−2.77	1,763.23	−0.01	1,763.03	0.14
400	100	nrr.	sm.	3	1,341.61	0.00	1,341.61	0.00	1,341.61	0.00
400	100	nrr.	sm.	10	1,277.65	0.03	1,276.99	0.00	1,278.05	−0.07
400	100	nrr.	sm.	100	1,272.61	0.02	1,272.10	−0.03	1,272.41	0.05
400	100	nrr.	lrg.	3	2,414.64	−2.23	2,174.86	−0.45	2,185.47	−0.78
400	100	nrr.	lrg.	10	2,863.89	−1.33	2,386.36	−0.83	2,392.44	−0.54
400	100	nrr.	lrg.	100	2,948.77	0.58	2,396.63	−0.27	2,396.55	1.41
400	100	wd.	sm.	3	902.05	0.17	917.81	−0.04	916.02	−0.67
400	100	wd.	sm.	10	1,036.20	−0.09	1,058.81	−0.75	1,085.47	−0.81
400	100	wd.	sm.	100	930.36	1.00	965.05	−0.82	993.41	−0.63
400	100	wd.	lrg.	3	2,422.14	−4.01	2,219.74	−1.13	2,219.95	0.56
400	100	wd.	lrg.	10	2,496.78	1.04	2,203.81	−0.11	2,284.59	0.63
400	100	wd.	lrg.	100	2,858.09	0.76	2,351.77	0.48	2,465.94	0.95
total					1,156.71	−0.01	1,081.24	−0.07	1,086.60	0.03

Table C.9: Comparison of loading approaches; extended problem variants (3L-VRPCB(TW), RL/SL, C1); average TTDs provided by rear loading (benchmark), average deviations of TTDs provided by side loading from benchmarks; separated by instance classes and variants of the hybrid algorithm

						ALNS					
						DBLF		**LS_DBLF**		**LS_OS**	
instance parameters						RL	SL	RL	SL	RL	SL
						∅TTD	∅dev	∅TTD	∅dev	∅TTD	∅dev
m	n	%LH	TW	it.	typ	[DU]	[%]	[DU]	[%]	[DU]	[%]
200	20	50	-	sm.	3	382.62	0.00	382.62	0.00	382.62	0.00
200	20	50	-	sm.	10	407.08	0.20	405.92	−0.07	405.67	0.06
200	20	50	-	sm.	100	400.09	0.20	399.00	0.00	400.58	−0.26
200	20	50	nrr.	sm.	3	514.56	0.00	514.56	0.00	514.56	0.00
200	20	50	nrr.	sm.	10	545.07	0.00	545.07	0.00	545.07	0.00
200	20	50	nrr.	sm.	100	496.81	0.01	496.81	0.00	496.81	0.00
200	20	50	wd.	sm.	3	470.23	0.00	470.23	0.00	470.23	0.00
200	20	50	wd.	sm.	10	483.35	0.00	483.35	0.00	483.35	0.00

Table C.9: Comparison of loading approaches; extended problem variants (3L-VRPCB(TW), RL/SL, C1); average TTDs provided by rear loading (benchmark), average deviations of TTDs provided by side loading from benchmarks; separated by instance classes and variants of the hybrid algorithm (*continued*)

						ALNS					
						DBLF		**LS_DBLF**		**LS_OS**	
instance parameters						RL	SL	RL	SL	RL	SL
						∅TTD	∅dev	∅TTD	∅dev	∅TTD	∅dev
m	n	%LH	TW	it.	typ	[DU]	[%]	[DU]	[%]	[DU]	[%]
200	20	50	wd.	sm.	100	479.44	0.37	479.39	0.00	479.39	0.00
200	20	80	-	sm.	3	374.82	0.28	374.08	−0.09	373.32	0.22
200	20	80	-	sm.	10	381.56	0.25	379.84	−0.22	382.48	−0.25
200	20	80	-	sm.	100	387.41	1.03	373.91	0.03	375.59	0.26
200	20	80	nrr.	sm.	3	475.82	0.00	475.82	0.00	475.82	0.00
200	20	80	nrr.	sm.	10	509.53	0.00	509.53	0.00	509.53	0.00
200	20	80	nrr.	sm.	100	503.50	0.66	500.30	0.00	500.30	0.00
200	20	80	wd.	sm.	3	418.35	−0.12	417.79	0.00	417.84	−0.01
200	20	80	wd.	sm.	10	401.91	0.44	395.19	0.03	400.22	−0.54
200	20	80	wd.	sm.	100	444.27	1.14	438.88	0.00	438.88	0.00
200	60	50	-	sm.	3	674.89	−0.28	673.39	0.40	674.00	0.37
200	60	50	-	sm.	10	683.78	−0.03	683.84	−0.05	688.67	0.17
200	60	50	-	sm.	100	669.79	0.15	670.01	−0.14	679.04	−0.43
200	60	50	-	lrg.	3	975.42	1.55	901.03	0.38	907.62	−0.56
200	60	50	-	lrg.	10	1,122.78	−0.77	930.38	−0.48	928.35	−1.02
200	60	50	-	lrg.	100	1,157.02	0.41	943.40	−0.14	929.54	−0.94
200	60	50	nrr.	sm.	3	1,045.73	0.00	1,045.73	0.00	1,045.73	0.00
200	60	50	nrr.	sm.	10	1,118.96	0.00	1,118.96	0.00	1,118.96	0.00
200	60	50	nrr.	sm.	100	1,193.37	0.00	1,193.46	−0.01	1,193.37	0.00
200	60	50	nrr.	lrg.	3	1,406.76	0.79	1,327.00	0.06	1,339.48	−0.91
200	60	50	nrr.	lrg.	10	1,301.63	−0.68	1,104.50	0.50	1,102.67	−0.65
200	60	50	nrr.	lrg.	100	1,304.55	2.69	1,120.90	−0.72	1,115.18	−1.16
200	60	50	wd.	sm.	3	903.93	0.00	903.93	0.04	904.23	−0.04
200	60	50	wd.	sm.	10	847.99	0.00	848.78	−0.13	847.99	0.08
200	60	50	wd.	sm.	100	865.34	−0.02	865.84	0.10	867.48	−0.32
200	60	50	wd.	lrg.	3	1,052.72	1.35	990.04	0.26	996.93	−0.31
200	60	50	wd.	lrg.	10	1,222.68	−1.12	1,033.14	−0.27	1,032.70	−0.57
200	60	50	wd.	lrg.	100	1,270.27	−0.47	1,062.36	0.87	1,053.35	−0.37
200	60	80	-	sm.	3	626.71	−0.02	630.55	0.23	627.29	0.81
200	60	80	-	sm.	10	619.01	0.58	627.82	0.17	635.96	−0.57
200	60	80	-	sm.	100	660.85	0.42	676.12	−0.50	683.17	0.33
200	60	80	-	lrg.	3	1,269.53	−4.68	1,142.46	−0.34	1,129.21	1.05
200	60	80	-	lrg.	10	1,204.04	−0.62	1,052.92	−0.80	1,053.08	0.15
200	60	80	-	lrg.	100	1,274.76	1.06	1,084.90	−0.39	1,073.47	0.65
200	60	80	nrr.	sm.	3	1,102.24	0.00	1,102.24	0.00	1,102.24	0.00
200	60	80	nrr.	sm.	10	1,133.31	0.00	1,133.31	0.00	1,133.31	0.00
200	60	80	nrr.	sm.	100	1,025.09	0.00	1,025.09	−0.13	1,025.09	0.00
200	60	80	nrr.	lrg.	3	1,508.03	−4.00	1,354.04	−0.51	1,345.41	0.77

Table C.9: Comparison of loading approaches; extended problem variants (3L-VRPCB(TW), RL/SL, C1); average TTDs provided by rear loading (benchmark), average deviations of TTDs provided by side loading from benchmarks; separated by instance classes and variants of the hybrid algorithm (*continued*)

						ALNS					
						DBLF		**LS_DBLF**		**LS_OS**	
instance parameters						RL	SL	RL	SL	RL	SL
						∅TTD	∅dev	∅TTD	∅dev	∅TTD	∅dev
m	n	%LH	TW	it.	typ	[DU]	[%]	[DU]	[%]	[DU]	[%]
200	60	80	nrr.	lrg.	10	1,314.72	−1.20	1,169.41	−0.43	1,161.58	−0.49
200	60	80	nrr.	lrg.	100	1,478.55	0.83	1,247.83	−1.79	1,217.89	0.58
200	60	80	wd.	sm.	3	718.21	0.03	718.26	−0.03	718.07	0.00
200	60	80	wd.	sm.	10	739.34	0.06	738.64	0.01	742.16	−0.21
200	60	80	wd.	sm.	100	836.52	−0.08	835.88	−0.07	837.71	0.08
200	60	80	wd.	lrg.	3	1,262.66	−5.98	1,124.85	0.83	1,114.51	2.17
200	60	80	wd.	lrg.	10	1,245.66	0.84	1,096.18	−0.98	1,089.82	−0.38
200	60	80	wd.	lrg.	100	1,354.57	1.13	1,138.21	0.29	1,125.96	−0.22
200	100	50	-	sm.	3	868.63	0.06	875.96	−0.02	874.47	0.42
200	100	50	-	sm.	10	869.14	0.15	875.74	0.26	892.80	−0.64
200	100	50	-	sm.	100	873.00	0.14	891.00	−0.59	918.42	0.19
200	100	50	-	lrg.	3	1,094.43	1.11	1,030.11	2.10	1,035.44	0.26
200	100	50	-	lrg.	10	1,212.60	−1.10	1,086.58	−0.23	1,086.87	−0.05
200	100	50	-	lrg.	100	1,298.97	−0.91	1,147.79	0.28	1,143.64	−0.48
200	100	50	nrr.	sm.	3	1,612.43	0.00	1,612.43	0.00	1,612.43	0.00
200	100	50	nrr.	sm.	10	1,448.39	−0.03	1,448.45	−0.04	1,447.95	0.00
200	100	50	nrr.	sm.	100	1,481.53	0.00	1,481.51	0.00	1,481.53	0.00
200	100	50	nrr.	lrg.	3	1,596.99	1.17	1,557.08	0.82	1,554.71	0.43
200	100	50	nrr.	lrg.	10	1,709.13	−0.43	1,575.65	0.22	1,582.47	−0.76
200	100	50	nrr.	lrg.	100	1,808.49	1.48	1,682.56	0.76	1,679.15	−0.24
200	100	50	wd.	sm.	3	1,092.62	0.08	1,092.50	0.09	1,092.62	0.04
200	100	50	wd.	sm.	10	1,209.26	0.06	1,210.13	0.00	1,210.09	0.00
200	100	50	wd.	sm.	100	1,247.02	0.14	1,249.47	−0.25	1,252.02	−0.30
200	100	50	wd.	lrg.	3	1,200.01	0.85	1,137.21	2.20	1,147.86	−0.30
200	100	50	wd.	lrg.	10	1,348.53	−0.90	1,254.85	−0.61	1,248.21	−0.12
200	100	50	wd.	lrg.	100	1,506.01	−0.28	1,370.59	−0.41	1,351.49	−0.05
200	100	80	-	sm.	3	842.76	−0.07	857.47	−0.16	855.48	−0.05
200	100	80	-	sm.	10	824.59	−0.05	846.30	0.32	860.49	−0.92
200	100	80	-	sm.	100	830.27	−0.01	868.26	−0.24	871.97	0.47
200	100	80	-	lrg.	3	1,236.36	2.06	1,209.82	0.65	1,214.11	0.19
200	100	80	-	lrg.	10	1,415.16	−2.56	1,302.44	−0.92	1,320.55	0.03
200	100	80	-	lrg.	100	1,449.72	−1.32	1,331.99	−0.69	1,357.82	1.80
200	100	80	nrr.	sm.	3	1,398.68	0.00	1,398.68	0.00	1,398.68	0.00
200	100	80	nrr.	sm.	10	1,401.89	0.00	1,401.89	0.00	1,401.97	0.00
200	100	80	nrr.	sm.	100	1,556.79	0.08	1,556.43	0.13	1,558.14	0.00
200	100	80	nrr.	lrg.	3	1,645.62	0.77	1,605.30	0.98	1,607.18	−0.29
200	100	80	nrr.	lrg.	10	1,620.30	−1.89	1,493.78	−0.16	1,488.34	−0.29
200	100	80	nrr.	lrg.	100	1,779.38	−1.79	1,636.31	−0.86	1,619.68	−0.14

Table C.9: Comparison of loading approaches; extended problem variants (3L-VRPCB(TW), RL/SL, C1); average TTDs provided by rear loading (benchmark), average deviations of TTDs provided by side loading from benchmarks; separated by instance classes and variants of the hybrid algorithm (*continued*)

						ALNS					
						DBLF		**LS_DBLF**		**LS_OS**	
instance parameters						RL	SL	RL	SL	RL	SL
						∅TTD	∅dev	∅TTD	∅dev	∅TTD	∅dev
m	n	%LH	TW	it.	typ	[DU]	[%]	[DU]	[%]	[DU]	[%]
200	100	80	wd.	sm.	3	1,132.00	0.00	1,132.00	0.01	1,132.00	0.00
200	100	80	wd.	sm.	10	1,000.08	0.04	999.34	−0.03	1,000.78	0.08
200	100	80	wd.	sm.	100	1,099.98	0.02	1,100.67	0.04	1,106.52	0.24
200	100	80	wd.	lrg.	3	1,302.25	1.15	1,270.70	0.48	1,269.25	0.49
200	100	80	wd.	lrg.	10	1,556.90	−2.34	1,435.81	−0.71	1,436.34	−1.64
200	100	80	wd.	lrg.	100	1,624.07	−0.91	1,503.62	−0.60	1,510.43	−0.46
400	20	50	-	sm.	3	414.03	0.48	411.04	−0.19	409.94	0.51
400	20	50	-	sm.	10	401.18	0.14	392.74	0.00	396.21	−0.09
400	20	50	-	sm.	100	431.91	1.27	416.60	−0.04	423.93	−1.10
400	20	50	nrr.	sm.	3	466.88	0.14	465.55	0.00	465.55	0.00
400	20	50	nrr.	sm.	10	609.15	0.06	607.17	0.00	607.17	0.00
400	20	50	nrr.	sm.	100	562.86	3.32	562.65	0.00	562.65	0.00
400	20	50	wd.	sm.	3	525.64	0.08	521.59	−0.03	521.43	−0.01
400	20	50	wd.	sm.	10	487.85	1.05	481.50	0.00	481.50	0.00
400	20	50	wd.	sm.	100	470.36	2.13	454.03	0.06	455.30	0.09
400	20	80	-	sm.	3	430.22	0.52	426.47	0.50	426.70	0.11
400	20	80	-	sm.	10	418.60	1.03	406.53	0.15	410.66	−0.03
400	20	80	-	sm.	100	455.80	3.26	427.03	1.21	444.76	−1.28
400	20	80	nrr.	sm.	3	623.67	0.05	618.60	0.00	616.10	0.00
400	20	80	nrr.	sm.	10	478.59	1.52	468.63	−0.41	469.98	0.20
400	20	80	nrr.	sm.	100	568.83	2.62	540.97	0.94	556.21	−1.07
400	20	80	wd.	sm.	3	419.19	0.76	418.33	−0.33	416.97	0.00
400	20	80	wd.	sm.	10	467.58	1.41	451.65	0.01	457.92	−1.09
400	20	80	wd.	sm.	100	500.42	1.48	477.52	−0.08	486.51	−0.61
400	60	50	-	sm.	3	698.21	0.07	699.07	0.48	702.08	−0.60
400	60	50	-	sm.	10	688.05	0.25	692.15	0.22	701.74	−0.16
400	60	50	-	sm.	100	686.04	0.55	688.45	−0.55	701.36	0.03
400	60	50	-	lrg.	3	1,591.85	−0.33	1,397.98	0.97	1,406.91	−0.37
400	60	50	-	lrg.	10	1,722.54	2.13	1,258.06	1.11	1,248.87	0.86
400	60	50	nrr.	sm.	3	1,087.51	0.00	1,087.51	0.00	1,087.51	0.00
400	60	50	nrr.	sm.	10	1,202.82	0.00	1,202.82	0.00	1,202.82	0.00
400	60	50	nrr.	sm.	100	1,146.57	0.06	1,145.98	0.05	1,145.98	0.00
400	60	50	nrr.	lrg.	3	1,860.95	0.46	1,727.25	0.95	1,740.86	−0.48
400	60	50	nrr.	lrg.	10	2,042.15	−5.68	1,461.02	1.53	1,469.23	−1.13
400	60	50	wd.	sm.	3	920.69	−0.07	919.18	0.00	919.36	0.07
400	60	50	wd.	sm.	10	776.42	0.12	782.62	−0.90	790.90	0.03
400	60	50	wd.	sm.	100	938.63	0.43	932.83	−0.11	949.00	1.10
400	60	50	wd.	lrg.	3	1,563.51	−0.78	1,357.69	0.21	1,354.24	0.31

Table C.9: Comparison of loading approaches; extended problem variants (3L-VRPCB(TW), RL/SL, C1); average TTDs provided by rear loading (benchmark), average deviations of TTDs provided by side loading from benchmarks; separated by instance classes and variants of the hybrid algorithm (*continued*)

						ALNS					
						DBLF		**LS_DBLF**		**LS_OS**	
instance parameters						RL	SL	RL	SL	RL	SL
						∅TTD	∅dev	∅TTD	∅dev	∅TTD	∅dev
m	n	%LH	TW	it.	typ	[DU]	[%]	[DU]	[%]	[DU]	[%]
400	60	50	wd.	lrg.	10	1,632.84	5.54	1,258.88	1.56	1,244.88	1.18
400	60	80	-	sm.	3	657.69	0.96	660.44	0.65	660.47	−0.02
400	60	80	-	sm.	10	669.98	0.32	684.92	−0.54	685.28	1.90
400	60	80	-	sm.	100	681.97	0.95	685.50	0.95	707.42	−1.01
400	60	80	-	lrg.	3	1,966.44	−0.45	1,865.28	−0.08	1,854.83	0.49
400	60	80	-	lrg.	10	2,146.25	−15.32	1,564.85	−1.87	1,554.91	−0.89
400	60	80	nrr.	sm.	3	975.59	0.00	975.59	0.00	975.59	0.00
400	60	80	nrr.	sm.	10	1,021.24	0.00	1,021.24	0.00	1,021.24	0.00
400	60	80	nrr.	sm.	100	1,001.59	0.06	1,000.68	0.00	1,000.60	0.00
400	60	80	nrr.	lrg.	3	2,117.01	0.58	1,981.50	1.35	1,988.68	−0.13
400	60	80	wd.	sm.	3	818.19	0.18	820.99	0.06	820.47	−0.05
400	60	80	wd.	sm.	10	834.27	−0.05	829.93	−0.13	841.11	−0.53
400	60	80	wd.	sm.	100	822.35	0.58	819.43	0.58	841.37	1.71
400	60	80	wd.	lrg.	3	1,576.88	−1.17	1,539.32	0.00	1,543.20	−0.28
400	100	50	-	sm.	3	878.18	−0.37	900.34	0.71	899.51	0.09
400	100	50	-	sm.	10	877.82	0.38	914.43	−0.53	914.59	0.94
400	100	50	-	sm.	100	897.47	−0.12	927.73	0.02	945.22	0.44
400	100	50	-	lrg.	3	1,831.84	−3.61	1,597.00	−0.19	1,586.07	0.63
400	100	50	-	lrg.	10	1,922.42	3.56	1,583.36	−0.90	1,580.97	−0.18
400	100	50	-	lrg.	100	2,020.74	3.59	1,595.67	0.28	1,589.71	0.46
400	100	50	nrr.	sm.	3	1,583.31	0.00	1,583.31	0.00	1,583.31	0.00
400	100	50	nrr.	sm.	10	1,498.96	−0.08	1,498.96	0.02	1,498.03	0.00
400	100	50	nrr.	sm.	100	1,446.99	0.09	1,446.99	0.05	1,446.99	0.05
400	100	50	nrr.	lrg.	3	2,077.80	−3.79	1,826.39	0.43	1,819.48	0.10
400	100	50	nrr.	lrg.	10	2,288.71	−0.71	1,898.81	−0.69	1,881.55	−0.37
400	100	50	nrr.	lrg.	100	2,420.57	2.15	1,969.91	−0.09	1,946.88	−0.26
400	100	50	wd.	sm.	3	1,122.85	0.06	1,130.06	−0.11	1,128.74	0.42
400	100	50	wd.	sm.	10	1,346.33	0.14	1,351.20	0.00	1,357.76	0.76
400	100	50	wd.	sm.	100	1,207.09	0.10	1,219.26	−0.13	1,247.38	0.90
400	100	50	wd.	lrg.	3	1,933.40	−2.11	1,743.41	−0.21	1,735.00	0.54
400	100	50	wd.	lrg.	10	1,896.58	3.20	1,580.72	−0.32	1,584.72	0.70
400	100	50	wd.	lrg.	100	2,151.32	6.00	1,744.93	−0.07	1,720.81	0.22
400	100	80	-	sm.	3	856.10	−0.29	882.28	−0.05	877.24	−0.39
400	100	80	-	sm.	10	856.25	0.02	886.97	−0.17	901.16	−0.32
400	100	80	-	sm.	100	877.93	0.70	905.04	1.43	943.71	−0.75
400	100	80	-	lrg.	3	1,824.76	−2.03	1,698.59	−1.19	1,721.12	−1.23
400	100	80	-	lrg.	10	2,104.02	3.07	1,832.56	0.61	1,882.99	1.21
400	100	80	-	lrg.	100	2,388.79	1.69	2,029.34	−0.87	2,091.24	1.81

Table C.9: Comparison of loading approaches; extended problem variants (3L-VRPCB(TW), RL/SL, C1); average TTDs provided by rear loading (benchmark), average deviations of TTDs provided by side loading from benchmarks; separated by instance classes and variants of the hybrid algorithm (*continued*)

						ALNS					
						DBLF		**LS_DBLF**		**LS_OS**	
instance parameters						RL	SL	RL	SL	RL	SL
						∅TTD	∅dev	∅TTD	∅dev	∅TTD	∅dev
m	n	%LH	TW	it.	typ	[DU]	[%]	[DU]	[%]	[DU]	[%]
400	100	80	nrr.	sm.	3	1,547.98	0.00	1,547.98	0.00	1,547.98	0.00
400	100	80	nrr.	sm.	10	1,512.81	−0.03	1,513.14	−0.06	1,512.81	0.00
400	100	80	nrr.	sm.	100	1,566.38	−0.01	1,565.92	0.00	1,565.32	0.06
400	100	80	nrr.	lrg.	3	1,981.70	1.75	1,880.75	−0.11	1,896.00	−1.11
400	100	80	nrr.	lrg.	10	2,626.04	−2.95	2,164.23	0.42	2,150.50	0.20
400	100	80	nrr.	lrg.	100	2,517.55	3.12	2,078.16	−0.13	2,072.74	0.52
400	100	80	wd.	sm.	3	1,115.52	0.45	1,135.32	0.03	1,132.79	−0.42
400	100	80	wd.	sm.	10	1,194.81	0.11	1,207.74	0.04	1,214.16	1.16
400	100	80	wd.	sm.	100	1,090.98	0.20	1,114.40	0.47	1,150.91	1.33
400	100	80	wd.	lrg.	3	1,986.59	−4.02	1,767.42	−1.76	1,780.92	−1.25
400	100	80	wd.	lrg.	10	2,069.98	2.68	1,825.95	0.76	1,941.14	−0.17
400	100	80	wd.	lrg.	100	2,357.51	1.81	2,007.06	−1.58	2,043.50	1.54
total						1,040.48	0.15	984.00	0.01	987.49	0.00

Table C.10: Comparison of loading approaches; extended problem variants (3L-VRPMB(TW), LSP/SL, C1); average TTDs provided by loading space partition (benchmark), average deviations of TTDs provided by side loading from benchmarks; separated by instance classes and variants of the hybrid algorithm

					ALNS					
					DBLF$^{(SL)}$		**LS_DBLF$^{(SL)}$**		**LS_OS**	
instance parameters					LSP	SL	LSP	SL	RL	SL
					∅TTD	∅dev	∅TTD	∅dev	∅TTD	∅dev
n	%LH	TW	it.	typ	[DU]	[%]	[DU]	[%]	[DU]	[%]
20	50	-	sm.	3	362.29	−10.08	351.52	−7.73	354.45	−9.77
20	50	-	sm.	10	376.57	−6.12	366.06	−4.14	370.57	−6.70
20	50	-	sm.	100	369.19	−2.02	357.60	−1.37	366.77	−5.46
20	50	nrr.	sm.	3	531.21	−17.55	495.40	−11.96	497.68	−12.34
20	50	nrr.	sm.	10	486.96	−6.90	477.86	−5.12	477.86	−5.37
20	50	nrr.	sm.	100	472.59	−4.96	466.90	−3.77	476.46	−6.33
20	50	wd.	sm.	3	406.24	−4.86	399.13	−3.14	399.88	−3.33
20	50	wd.	sm.	10	412.64	−2.59	392.57	1.85	399.14	−1.21
20	50	wd.	sm.	100	436.73	−1.27	417.17	2.50	425.08	−0.74
20	80	-	sm.	3	404.80	−14.33	391.52	−12.13	393.68	−12.83
20	80	-	sm.	10	391.22	−12.08	373.19	−10.24	377.74	−10.61
20	80	-	sm.	100	404.89	−8.58	378.19	−7.75	389.27	−6.96
20	80	nrr.	sm.	3	460.74	−4.20	453.55	−2.61	448.33	−1.96
20	80	nrr.	sm.	10	480.39	−1.71	474.98	−0.25	477.27	−0.90
20	80	nrr.	sm.	100	477.28	−3.24	464.95	−0.52	463.13	−0.51

Table C.10: Comparison of loading approaches; extended problem variants (3L-VRPMB(TW), LSP/SL, C1); average TTDs provided by loading space partition (benchmark), average deviations of TTDs provided by side loading from benchmarks; separated by instance classes and variants of the hybrid algorithm (*continued*)

					ALNS					
					DBLF$^{(SL)}$		**LS_DBLF$^{(SL)}$**		**LS_OS**	
instance parameters					LSP	SL	LSP	SL	RL	SL
					∅TTD	∅dev	∅TTD	∅dev	∅TTD	∅dev
n	%LH	TW	it.	typ	[DU]	[%]	[DU]	[%]	[DU]	[%]
20	80	wd.	sm.	3	439.15	−13.72	417.91	−10.23	422.93	−11.93
20	80	wd.	sm.	10	400.99	−11.91	382.98	−10.74	383.93	−10.19
20	80	wd.	sm.	100	414.44	−2.41	398.41	−2.29	399.70	−1.53
60	50	-	sm.	3	538.32	−3.45	544.27	−4.54	547.12	−6.49
60	50	-	sm.	10	549.46	−1.13	555.16	−1.93	561.76	−5.33
60	50	-	sm.	100	545.33	0.54	541.18	1.31	545.88	−0.82
60	50	-	lrg.	3	1,279.78	−36.23	1,112.77	−30.52	1,116.49	−31.26
60	50	-	lrg.	10	1,508.53	−37.51	1,145.46	−28.03	1,097.27	−26.13
60	50	nrr.	sm.	3	975.69	−8.32	903.99	−1.27	904.29	−1.31
60	50	nrr.	sm.	10	846.88	−5.84	804.87	−1.03	803.25	−0.81
60	50	nrr.	sm.	100	1,179.82	−14.04	1,045.16	−3.29	1,043.18	−3.05
60	50	nrr.	lrg.	3	1,450.20	−24.12	1,314.48	−18.49	1,287.20	−18.73
60	50	nrr.	lrg.	100	1,751.56	−27.07	1,625.35	−33.77	1,561.46	−32.26
60	50	wd.	sm.	3	730.96	−1.54	722.82	−0.41	725.44	−1.30
60	50	wd.	sm.	10	697.25	−1.36	700.18	−1.77	699.76	−2.67
60	50	wd.	sm.	100	696.12	−1.16	695.34	−1.03	691.75	−1.20
60	50	wd.	lrg.	3	1,245.89	−34.39	1,020.01	−24.63	1,005.40	−20.87
60	50	wd.	lrg.	10	1,347.73	−26.35	1,159.87	−26.94	1,097.35	−23.71
60	80	-	sm.	3	549.97	−5.35	545.84	−4.84	547.94	−5.63
60	80	-	sm.	10	544.24	−3.95	538.56	−2.76	540.30	−3.00
60	80	-	sm.	100	574.42	−3.86	562.29	−1.73	562.42	−0.44
60	80	-	lrg.	3	2,164.05	−38.30	2,048.41	−42.96	2,046.32	−42.77
60	80	-	lrg.	10	1,717.79	−39.57	1,503.71	−36.10	1,440.30	−33.38
60	80	-	lrg.	100	2,257.39	−45.20	1,797.35	−38.07	1,726.69	−36.77
60	80	nrr.	sm.	3	951.33	−0.47	946.54	0.00	946.54	0.00
60	80	nrr.	sm.	10	1,014.47	−1.93	994.05	0.00	994.05	0.00
60	80	nrr.	sm.	100	903.77	0.03	902.13	0.10	902.52	−0.05
60	80	nrr.	lrg.	3	2,094.04	−27.74	1,949.57	−30.03	1,948.40	−29.21
60	80	nrr.	lrg.	10	2,021.30	−39.77	1,715.13	−37.60	1,676.54	−36.45
60	80	wd.	sm.	3	607.60	−1.93	608.08	−2.15	605.04	−1.61
60	80	wd.	sm.	10	625.95	−1.35	617.61	0.30	622.74	−1.27
60	80	wd.	sm.	100	690.75	0.34	685.80	1.21	688.50	−0.26
60	80	wd.	lrg.	3	2,135.18	−42.21	2,017.49	−45.29	2,012.27	−44.13
60	80	wd.	lrg.	10	1,490.49	−34.25	1,318.94	−31.52	1,240.20	−27.11
60	80	wd.	lrg.	100	2,348.67	−41.65	1,830.65	−32.25	1,735.35	−30.27
100	50	-	sm.	3	680.50	−3.31	693.40	−4.50	689.35	−5.90
100	50	-	sm.	10	698.50	−2.81	716.00	−3.97	730.17	−6.50
100	50	-	sm.	100	698.95	−0.99	710.86	−2.15	722.16	−3.47

Table C.10: Comparison of loading approaches; extended problem variants (3L-VRPMB(TW), LSP/SL, C1); average TTDs provided by loading space partition (benchmark), average deviations of TTDs provided by side loading from benchmarks; separated by instance classes and variants of the hybrid algorithm (*continued*)

					ALNS					
					DBLF$^{(SL)}$		**LS_DBLF$^{(SL)}$**		**LS_OS**	
instance parameters					LSP	SL	LSP	SL	RL	SL
					∅TTD	∅dev	∅TTD	∅dev	∅TTD	∅dev
n	%LH	TW	it.	typ	[DU]	[%]	[DU]	[%]	[DU]	[%]
100	50	-	lrg.	3	1,624.74	−33.97	1,420.66	−27.32	1,382.89	−24.55
100	50	-	lrg.	10	1,788.09	−34.50	1,492.80	−25.46	1,431.24	−22.98
100	50	-	lrg.	100	1,866.26	−32.11	1,569.31	−24.02	1,517.76	−20.51
100	50	nrr.	sm.	3	1,526.31	−13.02	1,345.47	−1.81	1,345.47	−1.81
100	50	nrr.	sm.	10	1,381.54	−12.25	1,213.79	−1.57	1,213.58	−1.50
100	50	nrr.	sm.	100	1,385.64	−10.49	1,264.21	−2.49	1,264.72	−2.39
100	50	nrr.	lrg.	3	1,817.57	−21.91	1,580.52	−13.55	1,555.02	−12.31
100	50	nrr.	lrg.	10	2,127.91	−31.63	1,658.81	−16.81	1,576.25	−13.86
100	50	nrr.	lrg.	100	2,017.48	−18.01	1,695.93	−5.94	1,641.83	−3.33
100	50	wd.	sm.	3	904.60	−6.25	879.21	−3.72	882.96	−4.27
100	50	wd.	sm.	10	1,065.99	−9.99	1,000.82	−4.49	1,001.77	−4.62
100	50	wd.	sm.	100	1,036.00	−6.10	1,012.14	−4.06	1,020.84	−5.09
100	50	wd.	lrg.	3	1,615.94	−28.60	1,426.83	−21.69	1,394.81	−19.40
100	50	wd.	lrg.	10	1,903.19	−32.36	1,491.33	−17.99	1,425.72	−15.73
100	50	wd.	lrg.	100	2,032.34	−26.61	1,654.17	−15.80	1,588.03	−13.97
100	80	-	sm.	3	672.26	−1.98	682.75	−2.19	676.78	−3.20
100	80	-	sm.	10	681.29	−1.09	680.25	0.83	693.81	−1.34
100	80	-	sm.	100	696.88	−0.62	698.55	0.61	711.74	−1.67
100	80	-	lrg.	3	1,811.91	−35.02	1,739.85	−34.35	1,756.55	−35.32
100	80	-	lrg.	10	2,135.50	−35.63	1,953.49	−33.51	1,962.10	−33.24
100	80	-	lrg.	100	2,184.92	−33.28	1,953.76	−30.18	1,981.99	−30.15
100	80	nrr.	sm.	3	1,210.35	−0.19	1,211.04	−0.30	1,210.46	−0.24
100	80	nrr.	sm.	10	1,238.18	−0.42	1,233.59	−0.15	1,233.48	−0.18
100	80	nrr.	sm.	100	1,346.93	−0.33	1,342.74	0.00	1,342.26	0.00
100	80	nrr.	lrg.	3	1,940.43	−27.36	1,847.79	−26.44	1,833.98	−25.10
100	80	nrr.	lrg.	10	2,159.59	−27.03	1,935.67	−24.19	1,918.22	−24.39
100	80	nrr.	lrg.	100	2,324.52	−26.87	2,030.83	−21.65	1,957.73	−19.79
100	80	wd.	sm.	3	914.33	−0.23	917.13	−0.55	915.20	−0.29
100	80	wd.	sm.	10	804.38	−0.30	801.54	−0.04	808.02	−0.88
100	80	wd.	sm.	100	907.74	−0.11	905.94	0.34	914.33	−0.43
100	80	wd.	lrg.	3	1,825.05	−29.05	1,746.70	−28.39	1,745.29	−29.59
100	80	wd.	lrg.	10	2,333.37	−34.49	2,077.96	−30.37	2,068.87	−30.87
100	80	wd.	lrg.	100	2,372.86	−33.36	2,015.68	−26.36	2,029.19	−26.68
total					1,002.33	−11.77	931.92	−9.35	928.28	−9.80

Table C.11: Comparison of loading approaches; extended problem variants (3L-VRPSDP(TW), LSP/SL, C1); average TTDs provided by loading space partition (benchmark), average deviations of TTDs provided by side loading from benchmarks; separated by instance classes and variants of the hybrid algorithm

					ALNS					
					DBLF$^{(SL)}$		**LS_DBLF$^{(SL)}$**		**LS_OS**	
instance parameters					LSP	SL	LSP	SL	RL	SL
					∅TTD	∅dev	∅TTD	∅dev	∅TTD	∅dev
n	%LH	TW	it.	typ	[DU]	[%]	[DU]	[%]	[DU]	[%]
20	50	-	sm.	3	348.64	−5.18	338.60	−2.47	338.85	−5.20
20	50	-	sm.	10	368.64	1.66	353.88	5.75	357.09	−2.82
20	50	-	sm.	100	360.65	5.58	346.82	9.37	349.06	4.03
20	50	nrr.	sm.	3	450.21	−2.34	442.97	−0.66	442.10	−2.23
20	50	nrr.	sm.	10	462.75	0.54	462.75	0.54	462.75	0.00
20	50	nrr.	sm.	100	436.67	2.32	436.64	2.32	436.64	0.08
20	50	wd.	sm.	3	394.63	−1.62	392.14	−1.00	392.35	−1.50
20	50	wd.	sm.	10	394.84	2.87	389.62	3.44	390.32	0.05
20	50	wd.	sm.	100	416.53	2.92	413.19	3.71	413.19	0.85
20	80	-	sm.	3	402.98	−8.59	393.72	−6.78	392.82	−11.48
20	80	-	sm.	10	387.25	−3.19	375.34	−1.04	377.70	−7.48
20	80	-	sm.	100	404.93	−1.95	386.54	−0.71	392.83	−5.38
20	80	nrr.	sm.	3	450.31	−0.83	443.54	0.22	443.54	−0.96
20	80	nrr.	sm.	10	481.09	−0.41	475.88	0.83	475.88	−0.65
20	80	nrr.	sm.	100	464.63	0.87	458.91	1.04	458.91	−0.06
20	80	wd.	sm.	3	432.55	−9.33	418.12	−7.19	420.17	−11.34
20	80	wd.	sm.	10	395.94	−2.78	380.38	−1.93	383.39	−5.61
20	80	wd.	sm.	100	416.69	−0.03	402.07	0.53	406.07	−4.80
60	50	-	sm.	3	537.71	−2.53	532.47	−1.58	532.19	−3.61
60	50	-	sm.	10	547.64	1.24	544.95	1.69	545.79	−1.62
60	50	-	sm.	100	541.01	3.57	538.62	3.76	538.87	0.31
60	50	-	lrg.	3	1,433.96	−20.24	1,376.00	−18.11	1,367.00	−22.45
60	50	-	lrg.	10	1,543.12	−10.76	1,422.98	−6.31	1,397.09	−6.34
60	50	-	lrg.	100	1,540.01	−10.76	1,379.97	−4.17	1,342.39	−4.08
60	50	nrr.	sm.	3	892.22	0.00	892.22	0.00	892.22	0.00
60	50	nrr.	sm.	10	929.06	0.00	929.06	0.00	928.78	0.00
60	50	nrr.	sm.	100	997.61	0.05	997.61	0.04	998.02	0.00
60	50	nrr.	lrg.	3	1,523.39	−16.65	1,447.91	−13.49	1,431.45	−17.73
60	50	nrr.	lrg.	10	1,577.40	−16.26	1,460.68	−11.72	1,435.61	−10.94
60	50	nrr.	lrg.	100	1,576.23	−9.70	1,396.76	−2.44	1,347.33	1.24
60	50	wd.	sm.	3	718.75	0.00	718.65	0.05	718.73	−0.38
60	50	wd.	sm.	10	687.47	0.99	683.97	1.08	684.16	0.07
60	50	wd.	sm.	100	682.66	1.25	680.81	1.51	680.85	0.25
60	50	wd.	lrg.	3	1,450.28	−16.45	1,376.06	−13.39	1,365.38	−13.39
60	50	wd.	lrg.	10	1,578.01	−4.04	1,427.85	1.48	1,382.77	2.37
60	50	wd.	lrg.	100	1,607.79	−9.63	1,431.91	−1.93	1,385.07	−2.22
60	80	-	sm.	3	548.19	−4.05	546.94	−4.04	543.08	−4.68
60	80	-	sm.	10	542.60	−1.07	533.82	0.74	536.04	−2.77

Table C.11: Comparison of loading approaches; extended problem variants (3L-VRPSDP(TW), LSP/SL, C1); average TTDs provided by loading space partition (benchmark), average deviations of TTDs provided by side loading from benchmarks; separated by instance classes and variants of the hybrid algorithm (*continued*)

					ALNS					
					DBLF$^{(SL)}$		**LS_DBLF$^{(SL)}$**		**LS_OS**	
instance parameters					LSP	SL	LSP	SL	RL	SL
					∅TTD	∅dev	∅TTD	∅dev	∅TTD	∅dev
n	%LH	TW	it.	typ	[DU]	[%]	[DU]	[%]	[DU]	[%]
60	80	-	sm.	100	571.63	1.72	564.41	3.54	569.63	−1.08
60	80	-	lrg.	3	2,088.36	−33.02	1,966.33	−33.94	1,917.06	−33.66
60	80	-	lrg.	10	1,703.41	−28.70	1,537.03	−26.92	1,494.13	−27.86
60	80	-	lrg.	100	1,905.06	−25.04	1,710.91	−24.24	1,653.10	−25.56
60	80	nrr.	sm.	3	946.54	0.00	946.54	0.00	946.54	0.00
60	80	nrr.	sm.	10	994.39	−0.04	994.05	0.00	994.05	0.00
60	80	nrr.	sm.	100	902.13	0.19	902.13	0.14	902.13	0.00
60	80	nrr.	lrg.	3	2,072.95	−29.31	1,902.29	−25.91	1,845.46	−24.23
60	80	nrr.	lrg.	10	1,743.76	−24.52	1,553.04	−21.12	1,495.33	−22.51
60	80	nrr.	lrg.	100	1,881.48	−24.20	1,707.14	−23.92	1,624.28	−24.60
60	80	wd.	sm.	3	608.08	−1.79	604.25	−1.20	603.66	−1.48
60	80	wd.	sm.	10	622.41	−0.46	620.36	−0.16	620.34	−0.68
60	80	wd.	sm.	100	689.59	1.28	685.20	2.20	686.47	−0.39
60	80	wd.	lrg.	3	2,113.60	−31.89	2,000.28	−33.91	1,946.61	−35.03
60	80	wd.	lrg.	10	1,742.56	−27.08	1,568.38	−23.51	1,532.36	−26.83
60	80	wd.	lrg.	100	1,940.77	−22.60	1,739.40	−21.28	1,689.38	−21.83
100	50	-	sm.	3	670.05	−0.54	669.95	−0.37	671.30	−2.44
100	50	-	sm.	10	678.02	1.61	679.28	2.07	693.95	−1.27
100	50	-	sm.	100	685.30	3.23	687.84	3.08	706.02	−1.11
100	50	-	lrg.	3	1,594.82	−16.33	1,547.33	−14.38	1,556.09	−15.80
100	50	-	lrg.	10	1,677.64	−16.40	1,606.22	−13.94	1,627.09	−17.39
100	50	-	lrg.	100	1,764.99	−14.82	1,677.65	−12.40	1,707.16	−13.85
100	50	nrr.	sm.	3	1,320.49	0.02	1,320.49	0.02	1,320.49	0.00
100	50	nrr.	sm.	10	1,194.25	0.02	1,194.73	−0.07	1,194.55	0.00
100	50	nrr.	sm.	100	1,233.67	0.07	1,232.81	−0.01	1,233.67	0.00
100	50	nrr.	lrg.	3	1,736.99	−3.18	1,641.07	1.86	1,632.53	3.75
100	50	nrr.	lrg.	10	1,898.79	−10.27	1,793.70	−6.83	1,750.38	−5.81
100	50	nrr.	lrg.	100	1,927.80	−8.81	1,809.75	−5.07	1,747.89	−4.70
100	50	wd.	sm.	3	851.32	−0.56	850.02	−0.26	848.10	−0.38
100	50	wd.	sm.	10	956.14	0.05	956.01	−0.13	955.98	0.05
100	50	wd.	sm.	100	970.25	0.30	968.53	0.36	969.52	−0.09
100	50	wd.	lrg.	3	1,619.03	−18.31	1,573.92	−17.09	1,560.19	−19.71
100	50	wd.	lrg.	10	1,621.22	−13.43	1,543.72	−10.26	1,517.43	−13.39
100	50	wd.	lrg.	100	1,842.45	−12.45	1,725.83	−8.27	1,727.46	−9.55
100	80	-	sm.	3	674.59	−1.98	678.67	−0.68	678.11	−3.05
100	80	-	sm.	10	682.90	0.23	686.14	2.19	696.11	−1.13
100	80	-	sm.	100	699.85	0.46	703.36	2.80	708.94	1.06
100	80	-	lrg.	3	1,961.31	−33.63	1,908.29	−33.00	1,937.49	−35.49

Table C.11: Comparison of loading approaches; extended problem variants (3L-VRPSDP(TW), LSP/SL, C1); average TTDs provided by loading space partition (benchmark), average deviations of TTDs provided by side loading from benchmarks; separated by instance classes and variants of the hybrid algorithm (*continued*)

					ALNS					
					DBLF$^{(SL)}$		**LS_DBLF$^{(SL)}$**		**LS_OS**	
instance parameters					LSP	SL	LSP	SL	RL	SL
					∅TTD	∅dev	∅TTD	∅dev	∅TTD	∅dev
n	%LH	TW	it.	typ	[DU]	[%]	[DU]	[%]	[DU]	[%]
100	80	-	lrg.	10	2,216.77	−31.19	2,078.47	−30.01	2,128.93	−31.38
100	80	-	lrg.	100	2,159.60	−28.70	2,033.48	−27.52	2,078.97	−28.24
100	80	nrr.	sm.	3	1,208.49	0.00	1,207.51	0.04	1,207.96	0.00
100	80	nrr.	sm.	10	1,231.87	0.00	1,232.45	0.03	1,232.28	0.00
100	80	nrr.	sm.	100	1,342.20	0.08	1,342.71	−0.04	1,341.30	0.00
100	80	nrr.	lrg.	3	2,135.80	−30.52	2,060.16	−29.75	2,067.06	−29.67
100	80	nrr.	lrg.	10	2,249.42	−24.01	2,096.48	−22.70	2,075.78	−24.52
100	80	nrr.	lrg.	100	2,246.55	−21.87	2,059.73	−19.44	2,015.83	−19.94
100	80	wd.	sm.	3	912.65	0.09	912.65	0.16	912.81	−0.02
100	80	wd.	sm.	10	802.32	0.00	801.67	−0.02	802.77	−0.17
100	80	wd.	sm.	100	909.43	0.13	907.14	0.73	916.78	−0.52
100	80	wd.	lrg.	3	1,909.89	−28.01	1,836.65	−27.14	1,834.81	−29.57
100	80	wd.	lrg.	10	2,320.89	−30.38	2,160.48	−29.23	2,191.80	−30.93
100	80	wd.	lrg.	100	2,274.85	−25.31	2,134.53	−24.72	2,160.28	−26.33
total					1,095.75	−7.81	1,049.29	−6.29	1,045.83	−8.16