universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Urban Two Echelon Inventory Routing Problem for perishable goods with a waste reduction objective“

verfasst von / submitted by

Rebekka Prader BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2022 / Vienna, 2022

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066 915
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Betriebswirtschaft
Betreut von / Supervisor:	Univ.-Prof. Mag. Dr. Karl Franz Dörner, Privatdoz.

Abstract

In this master thesis a two-echelon inventory routing system accounting for the impact of transportation, inventory holding and waste of a perishable item is modeled. The presented mathematical formulation is designed in collaboration with a local producer and deliverer of organic horticultural products in the area of Vienna. For its solution a matheuristic resorting to two ALNS procedures and a reduced mathematical model is proposed.

The newly generated problem includes a large central depot outside of the city center, where harvested products are stored in the first place. A set of large combustion-engine vehicles delivers certain smaller distribution centres in different locations in the city, which serve as starting point for the second level routes performed by cargo bikes and delivering final customers. Inventory quantities at the central depot and the distribution centers are optimized together with the delivery routes on both echelons. Additionally, the gradual decease of the perishable good is reflected by the model. The objective is to minimize (environmental) costs of transportation, storage and waste while satisfying all customer demand.

Kurzfassung

In dieser Masterarbeit wird ein Inventory Routing Problem auf zwei Ebenen entworfen, das den Einfluss von Transport, Lagerhaltung und anfallendem Abfall eines verderblichen Gutes betrachtet. Das vorgestellte mathematische Modell wurde in Kollaboration mit einem lokalen Erzeuger und Zulieferer von biologischen Lebensmitteln im Liefergebiet Wien entworfen. Zur Lösung des Problems wird eine Matheuristic, die auf die Lösung einer reduzierten Version des Problems zurückgreift und zwei verschiedene ALNS Algorithmen inkludiert, vorgestellt.

Das entworfene Modell beinhaltet ein Hauptlager außerhalb des Stadtzentrums, in dem Produkte nach der Ernte eingelagert werden. Große Fahrzeuge mit Verbrennermotor beliefern von dort aus mehrere kleinere Zwischenlager, die als Startpunkt für die Routen zum Endkunden dienen, welche von Lastenfahrrädern gefahren werden. Die Lagermengen im Hauptlager und in den Zwischenlagern werden gemeinsam mit den Lieferrouten auf beiden Ebenen optimiert. Zudem wird der schrittweise Verfall des verderblichen Gutes modelliert. Die Zielfunktion setzt sich aus der Minimierung von Liefer-, Lager- und Abfallkosten zusammen, wobei die gesamte Nachfrage der Kunden erfüllt werden muss.

Contents

List of Abbreviations

ALNS	adaptive large neighbourhood search
ALNS-DP	adaptive large neighbourhood search to change delivery patterns
ALNS-2LR	adaptive large neighbourhood search to change 2nd level routing
COVID 19	Coronavirus disease 2019
CVRP	capacitated vehicle routing problem
CTD	close to deterioration
DC	distribution center
GHG	greenhouse-gases
IRP	inventory routing problem
MILP	mixed integer linear program
PIRP	perishable inventory routing problem
VRP	vehicle routing problem
2E	two echelon

List of Tables

List of Figures

1 Introduction

Food waste, especially if it happens post-harvest, is a global problem with extreme environmental impacts. Yahia et al. (2019) state that for some horticultural products in some regions and seasons the proportion of such postharvest loss can be up to 60% of the harvested quantity. According to the "preparatory study on food waste across EU 27" in the European Union 179 kilograms of food were wasted in 2010 per person on average (European Union, 2010). Overproduction and overstocking, as well as inefficiencies in the coordination between different stages of the supply chain are named as some relevant factors contributing to this extremely large number.

The environmental damage caused by post-harvest loss for perishable food items is manifold. Next to the waste of resources as land, water, chemicals and energy for the production, storage and transportation cause the emission of greenhouse-gases (GHGs) and other negative externalities as well. (Yahia et al., 2019) For all these reasons, the United Nations defined halving the "per capita global food waste at the retail and consumer level and reduce food losses along production and supply chains, including post-harvest-losses" by 2030 as one of their Sustainable Development Goals.

The presented master thesis aims at designing an inventory routing system accounting for the impact of transportation, inventory holding and waste of a perishable item. The model is inspired by and designed in collaboration with a local producer and deliverer of organic horticultural products in the area of Vienna. More precisely, our business partner grows fruits and vegetables on the outskirts of Vienna, stores them in a large depot outside of the city and delivers households and offices with boxes of those highly perishable goods. Such a company faces a range of different factors that must be optimized simultaneously: next to transportation and storage, here a large cost factor are the costs of overproduction and food wastage, which is a central subject of this work.

With regard to the costs of transportation, the local production of our business partner already provides a higher closeness to customers compared to international supply chains. However, last-mile delivery in an urban context is often considered particularly inefficient (Gevaers et al., 2009). Additionally, the extreme expansion of e-commerce and home deliveries in the past years and the connected increase of delivery vehicles in urban areas leads to a considerable amount of congestion, GHG emission, noise, and health issues (Boysen et al, 2021).

Another problem encountered in inner-city delivery are traffic limitations and one-way streets which might require large de-tours. To avoid these troubles, the presented model falls back on cargo bikes for home deliveries of horticultural products, which have no considerable contribution to noise, GHG emission and can even reduce congestion by using bike lanes. Especially in residential areas this type of vehicles might be more flexible than combustion-engine vehicles because they are not affected by traffic limitations and

can use one-way streets in both directions. Also our business partner already resorts to cargo bikes for some areas of Vienna for these reasons. However, like in Anderluh et al. (2016) and Anderluh et al. (2019), to counteract the load and distance limitations connected to the use of cargo bikes, the presented transportation system is organized on two echelons. Strictly speaking, we assume that there is a large depot outside of the city where the harvested agricultural products are stored in the first place. A set of large combustion-engine vehicles delivers certain smaller distribution centres (DCs) in different locations in the city, which serve as starting point for the second level routes performed by cargo bikes.

Such a two-echelon transportation system with intermediate storage of products is common in urban last-mile delivery. The presented model assumes that customers are available only in certain periods of the planning horizon and the company decides when to supply them. Inventory quantities at the large terminal and the DCs are optimized together with the delivery routes on both echelons. The objective is to minimize (environmental) costs for transportation, storage and waste while satisfying all customer demand.

The present study was originally motivated by the fact, that our business partner faced a constant demand increase in the past years as well as a sudden demand explosion at the beginning of the COVID-19 crisis. This growing relevance of online sales of fruits and vegetables was also outlined in the study about "Online sales of fruit and vegetables in Europe" conducted by Freshfel Europe and the OECD Fruit and Vegetables Scheme. It found, that especially the share of agricultural suppliers which deliver customers directly, was increasing considerably in the past years and that this development was additionally reinforced by the pandemic. For instance, based on Google Trends, the worldwide Google searches for the keywords "local food" and "food delivery" were higher than ever before in April 2020 (Shveda, 2020). Therefore, the model presented in this master thesis integrates different aspects that are relevant for a business model that seems to be of growing relevance.

The remainder of this master thesis is structured as follows: first, the literature on related topics is summarized, followed by a detailed problem description and the presentation of the novel two echelon inventory routing problem with waste-related objectives. Since the problem is np-hard and optimally solvable only for very small instances, a reduced version of the mathematical formulation is presented afterwards, which is the starting point for a matheuristic that generates second level routes resorting to two different Adaptive Large Neighbourhood Search (ALNS) algorithms. In the computational study, the model is tested on some well known and several newly generated instances.

2 Literature Review

Since the Vehicle Routing Problem (VRP), a generalized Traveling Salesman Problem, was introduced by Dantzig and Ramser (1959) it has been studied extensively. This is not only due to the fact, that it can be applied to an extremely broad range of practical issues, but also because of its considerable complexity.

In general, VRP stands for a class of problems in which a fleet of vehicles must be routed in a cost minimizing way to fulfil the demands of a given set of customers. This can be necessary not only in transportation and supply chain management but also in production planning, telecommunication and many more fields. (Toth & Vigo, 2014)

The intensive studies of the VRP in the past decades lead to the development of multiple variants of the problem. One of them is the Inventory Routing Problem (IRP), which answers different questions arising in supply chains, simultaneously. Here, the supplier does not only decide on when and how to deliver customers but also aims at optimizing the inventory levels of the given storage locations. Since the general IRP was studied intensively since more than thirty years, we want to refer to the reviews of Coelho and Laporte (2013), Coelho et al. (2014) and to the book by Bertazzi and Speranza (2013) for a general overview, while some influential newer works will be mentioned below.

The most relevant literature for the present master thesis, involving inventory routing of perishable products and environmental considerations, shall be introduced in the following sections.

2.1 Literature regarding storage and distribution of perishable products

Waste reduction in a supply chain concerned with perishable products can be viewed from different perspectives. While many peers developed waste reducing policies based on production and location decisions, the focus here lies on inventory and transportation. There are of course many papers which examine one of these fields individually. For instance, Tarantilis and Kiranoudis (2001), Amorim and Almada-Lobo (2014) and Song and Ko (2016) studied VRPs for perishable products with different considerations. A VRP with Time Windows for perishable goods is studied in Hsu et al. (2007), Osvald and Stirn (2008), Rabbani et al. (2015), Wang et al. (2016), to name only a few. Rabbani et al. (2016) include several intermediate depots and design a genetic algorithm to solve a single period case with a “refreshment” option for products. The age of the transported goods is not tracked precisely in this work but the level of decay only depends on the route length instead.

As already mentioned, all these papers do not explicitly include inventory considerations in the developed models. In contrast to a simple VRP, the IRP integrates inventory management as well as routing into the decision, which is especially important for perishable goods that are highly sensitive to storage and delivery times. Coelho et al. (2014) define the IRP as a problem in which the supplier decides about the timing, quantity and routes for customer delivery. Some influential newer works in the field of IRPs are, for instance, the one by Bertazzi et al. (2019), that introduces the multi-depot IRP as a new problem variant and solves it with a three-phase matheuristic. This is extended in Coelho et al. (2020) to a multi-attribute IRP by integrating a heterogeneous set of vehicles as well as multiple products. Alinaghial et al. (2020) present a green inventory-routing problem with time windows and solve it with a Tabu Search algorithm. However, since the literature on IRPs with different considerations is vast, the remainder of this section will focus on those papers in this context that include perishability considerations and are therefore most relevant for this work. In academic literature, those are often referred to as perishable inventory routing problems (PIRPs), see e.g. Shaabani (2022).

An early example here is the one by Federgruen et al. (1986) where a central depot delivers several regional centres with random demand. Depending on the age of the product it is classified as "fresh" or "old" and wasted at a certain age. Hiassat and Diabat (2011) additionally integrated a location decision into the model. Six years later, Hiassat et al. (2017) again developed an integrated location-inventory-routing problem, solving it by a genetic algorithm. This study was based on the one by Le et al. (2013), which generated a solution method based on cutting planes and column generation. Al Shamsi et al. (2014) consider a product with a fixed lifetime and do not allow for it to be wasted. Like in the present master thesis, CO_2 emission for transportation is included in the objective function. Coelho and Laporte (2014) developed a branch and cut algorithm to solve a perishable IRP with different selling policies to optimality. Additionally, they integrate an age tracking approach for the perishable good. In Jia et al. (2014) a two-phase algorithm is applied to solve a periodic IRP with shelf-life limits. This study is one of the first to integrate loading costs into the objective functions in this context.

A paper that is very relevant for the present master thesis is the one by Soysal et al. (2015), which deals with environmental aspects in a stochastic IRP with perishable products. They also include costs for routing, waste, and fuel consumption into the objective function. The quality of the resulting chance constrained problem is evaluated by a simulation model. However, in contrast to the present master thesis, this work does not include a 2E system and no deterministic solution method is designed. Azadeh et al. (2017) developed a genetic algorithm to solve an IRP with transhipment for a deteriorating good. Another genetic algorithm was presented by Rahimi et al. (2017) for a periodic IRP with multiple objectives including customer satisfaction, profit, and environmental impact. Crama et al. (2018) present different solution algorithms for a stochastic IRP with service levels for perishable products on a single level and compare their performance. There are also some recent papers that deal with perishable goods in agri-food supply chains. For example, Onggo et al. (2019) consider a single supplier serving various intermediate retailers in a PIRP with stochastic demand and solve it by a

iterated local search including a Monte Carlo simulation. Voli et al. (2020) consider a multistage stochastic IRP with deteriorating products in the agri-food sector. As in the other papers mentioned so far, no second level is considered here. The problem is solved my a rolling horizon approach.

In the present work, a special case of the classical IRP is applied, that is inspired by the work of Rohmer et al. (2019). While in most IRP literature there is no customer order at all but the supplier decides about when and how much to deliver, in our case a simplified version of this will be applied: We assume a fixed delivery quantity and number of deliveries per planning period for each customer. The supplier in this case can only decide about the timing of delivery within a set of acceptable delivery days of the customer. Put differently, the inventory decision taken here is the selection of the delivery day. This means that the supplier manages the inventory by delivering in a way that minimizes waste quantity and thus the waste costs.

2.2 Literature regarding two-echelon problems and perishability

As already mentioned, it is very common to organize urban operations in a two-level fashion. Usually this means that large vehicles with higher environmental costs deliver some intermediate depots, where storage takes place until smaller delivery vehicles carry the goods to final customers. Two Echelon (2E) VRPs have been studied by many peers. For a general overview we refer to the review by Cuda et al. (2015), while here only the most relevant works are described briefly.

Some basic models and heuristics for 2E VRPs and Capacitated VRPs (CVRPs) are introduced by Perboli et al. (2011). Hemmelmayr et al. (2012) designed an ALNS with small and large impact destroy operators to solve a similar problem. While small impact operators affect only the routes on the second level, the latter mentioned change the solution considerably by destroying some first level elements. Breunig et al. (2019) studied a 2E VRP with the use of electric vehicles on the second level.

Compared to the 2E VRP, the 2E IRP is a relatively novel problem class that was not studied by many peers so far. Zhao et al. (2008) propose an inventory routing problem on three echelons but without the consideration of perishability. A variable large neighbourhood search heuristic is used to solve the problem. Rohmer et al. (2019) developed a 2E IRP including a single supplier which delivers a single depot. On the second level, one of several possible delivery patterns must be chosen for every customer and their delivery routes are optimized. A matheuristic is proposed in which the first echelon is solved by a reduced model and a ALNS procedure is applied for the second echelon. Guimarães et al. (2019) implemented a vendor managed inventory policy where the intermediate depots control pickups from the suppliers and deliveries to the end user. A branch and cut algorithm is applied to solve the problem to optimality and a rich matheuristic is proposed for the solution of large instances. Farias et al. (2020) present a mathematical model for a 2E IRP including different inventory policies. They do not propose an algorithm for solving large instances. Schenekemberg et al. (2020) integrate

fleet management into a problem without perishability considerations. The same authors (2021) also developed a 2E production-routing problem that was solved using a branch and cut algorithm for different inventory strategies. As already mentioned, the focus of this paper lies on inventory and transportation though, not on production and location routing.

2.3 Contribution

As the literature review shows, there are many research gaps in the field of 2E IRPs. Even though there already exists some research regarding agri-food supply chains on a single level in different versions, not many papers of them include two echelons as commonly applied in urban transportation. One attempt to combine these issues is the paper of Rohmer et al. (2019). The novel model presented in this master thesis is based on this paper and takes up many of the included aspects, as the age-dependent deterioration approach and the two level organization scheme. However, it extends it by the following assumptions (for more details please see chapter 3):

- ⋆ The age of each unit of the good is modeled in a more complex fashion. To be precise, each unit enters the system with an age of zero and its age increases with each day of one. Goods are wasted at a certain age that is fixed. As in Rohmer et al. (2019) a gradual quality decrease is included in the holding costs, but there is an additional penalty for deteriorated products.
- ⋆ Not a single intermediate depot is considered but multiple ones.
- ⋆ The objective function is made more complex in order to integrate environmental considerations and especially waste reduction.
- ⋆ The matheuristic from the original paper is enriched to fulfil the requirements of the extended problem, especially by integrating a feedback loop.

This master thesis, therefore, contributes by presenting a 2E IRP with realistic assumptions that can be applied to real-life local food supply chains as the one of our business partner. The complex objective function is designed to reduce negative externalities of urban last mile delivery together with post-harvest waste of horticultural products.

3 Problem Formulation

In this master thesis, an urban last-mile delivery problem for perishable products is considered. As already mentioned, we assume a two-echelon inventory routing setting. Harvested horticultural products are first stored in a large depot, the major terminal, and delivered to several smaller distribution centres (DCs) by a set of large combustion-engine vehicles with high loading capacity. Note that there is a storage capacity limit for the DCs but not for the major terminal.

It is assumed that, for environmental reasons and traffic limitations, consolidation is mandatory, thus first level vehicles cannot supply customers directly. Like in the paper by Farias et al. (2020) the assignment of customer to DCs is given. This is reasonable if cargo bikes, which have limited reach and a relatively low loading capacity, supply the customers. Our business partner, for instance, delivers the city of Vienna district-wise to simplify operations.

3.1 Illustrative example

The five pictures in 3.1 show sample first-level routes for an example instance with a single vehicle available at the major terminal.

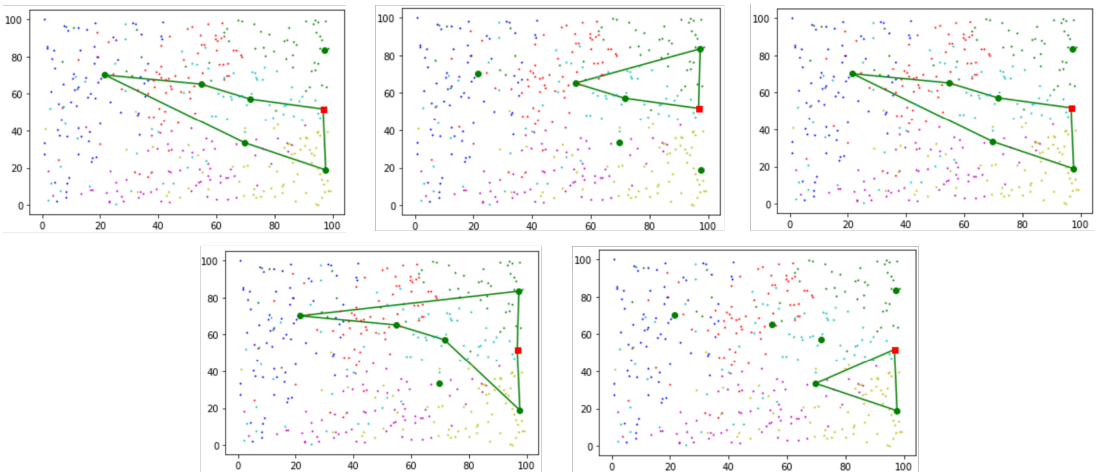

Figure 3.1: Randomly generated instance with six DCs (green circles), one terminal (red square) and N=400 customers, represented by the small dots, where all customers assigned to a specific DC share the same color. Each picture depicts the first level tour at one of the five days of the planning period.

As this example shows, not all DCs are delivered on each day of the week, which saves transportation costs, but might influence holding costs since like this the demand of the concerned day must already be on stock. However, when deciding about when and with which quantity to deliver each DC, many of the decisions taken on the second echelon have a strong influence.

3.2 Second echelon decision factors

3.2.1 Delivery day assignment

As indicated in previous chapters, a set of acceptable delivery days is assigned to each final customer and the supplier is responsible for choosing on which of the allowed days to deliver. Some customers are supplied once, others more than once in the planning horizon with a fixed demand quantity. Such simple delivery patterns give some flexibility to the seller to deliver some customers earlier if a high number of close-to-deterioration (CTD) products are on stock. Of course, the decision about when to deliver customers has a strong impact on first level routes as well, since the amount that is held on stock in the DCs must suffice to fulfil the resulting demand on the second level and we want to avoid a case, in which each DC must be delivered on each day of the planning period. The example in 3.1 shows an assignment of feasible delivery days to five customers that must be delivered either once or twice per planning period.

Customer	Day 1	Day 2	Day 3	Day 4	Day 5
A (1)		•			
B (2)				•	•
C (1)		•			
D (2)			•		•
E (1)					•

Table 3.1: Feasible delivery pattern selection for customers A to E in a planning horizon of five days. Grey cells represent days on which a customer is unavailable while white cells stand for possible delivery days. The number in parenthesis after each customer stands for the number of required deliveries while the black circles mark chosen delivery days.

If we assume, that customers A and B are assigned to DC 1 and C, D and E to DC 2, we get a completely deterministic case where the demand of DC 1 on day 2 is equal to the given demand of customer A on this day. Equally, the demand of DC 2 on day 5 is the sum of the given demand of customers D and E. This example proves, that as soon as the delivery day assignment decision is made, generating the second level routes breaks down to the solution of several independent VRPs, one for each DC on each day. We get a case with fixed and deterministic customer demands for each day, the assignment of customers to DCs is fixed as well and supply will always suffice since missing quantities

can be outsourced and delivered to customer on the same day. Figure 3.2 shows the tours performed from the DC on the bottom right of the example instance from the previous section at a specific day. As soon as we know which customers to serve on this day of the week, building delivery routes is nothing more than the solution of a VRP with full information, independent from the other decisions. This is a fact that is crucial for the matheuristic that will be presented in chapter 5.

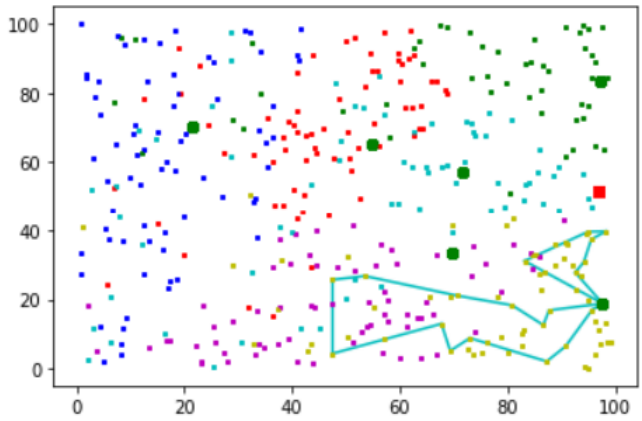

Figure 3.2: Tours performed to deliver all customer which are assigned to the DC at the bottom right and are chosen for delivery at the depicted day.

3.2.2 Deterioration modelling

One additional decision that is crucial for problems with perishable goods is how deterioration is modelled. One option here is integrating a fixed expiry date up to which delivery is possible. This can be the case for dairy products or meat in many cases and is discussed in more detail by Nahmias (2011). On the other hand, for many horticultural products as fruits and vegetables deterioration occurs gradually. We model this similarly to Rohmer et al. (2019) by tracking the age of every product precisely and penalizing the supply of final customers with CTD products by including age-dependent holding costs into the model. However, since waste reduction is a central objective of our business partner as well, we additionally penalize products which reach their maximum lifetime. They are wasted in the storage location that they are currently stored in and penalty costs for food wastage are incurred. We assume, that no deteriorated produce is transported to the final customer and that the maximum lifetime is known and given.

3.3 Objectives

In general, the presented model includes some elements from Rohmer et al. (2019) as the two echelon setting, the age-dependent holding costs and the interpretation of inventory routing as a selection of delivery patterns. However, while Rohmer et al. only include a

single DC in their model, we use multiple DCs that are spread over the delivery area. A mathematical formulation for such a 2E IRP with multiple DCs was generated by Farias et al. (2020). We extend their formulation by several aspects, as a more complex objective function as well as waste and deterioration considerations. Our objective function is based on the one in Soysal et al. (2015), integrating the emission model for urban transportation from Ehmke et al. (2018). It simultaneously chooses delivery patterns for each end customer, determines first and second level routes and optimizes the stock level in each DC. Customer demands for the planning horizon of one week are assumed as known and given. The same is true for the supply of home-grown horticultural products because the time span between plantation and harvesting is longer than the planning period and therefore supply quantities cannot be changed short-term. However, if the supply of self-produced horticultural products does not suffice to fulfil all customer demand, outsourcing is allowed in the form of orders from different producers.

4 Mathematical Model

4.1 Sets and Parameters

Sets	
V	set of all nodes (terminal, DCs, customers), V = $V_0 \cup V_{DC} \cup V_C$
V_0	{0} = major terminal/central depot
V_{DC}	{1, ..., $\|V_{DC}\|$} = distribution centres (DCs)
$V_{C,u}$	subset of customers assigned to u $\in V_{DC}$
V_C	{$\|V_{DC}\|+1, \ldots, \|V\|$} = final customers , supplied only by the DCs (not V_0)
K_e	set of all vehicles that are available on levels e = {1,2}
K_1	{1, ..., b_1} = vehicles on the first level (departing from the terminal)
K_2	{1, ..., b_2} = vehicles on the second level (departing from the DCs)
T	{1, ..., t} = planning horizon
G	age of a product g = {0, 1, ..., m-1}, deteriorates at age m
Parameters	
C_i	inventory capacity of node i $\in V_0 \cup V_{\mathrm{DC}}$
$I_i^{0,g}$	quantity stored at node i $\in V_0 \cup V_{\mathrm{DC}}$ of age g $\in$ G at time t=0
R_e	vehicle capacity for the vehicles of echelons e = {1, 2} where $R_2 << R_1$
b_e	number of vehicles at terminal in e = {1} and at each DC in e = {2}
D_i	set of days on which a customer i $\in V_C$ is available for delivery
c_t	order costs for externally procured products in period t $\in T$
P	maximal order quantity per period
s_t	supply (locally grown products) in period t $\in T$
d_i	demand of customer i $\in V_C$
o_e	wage rate of drivers on both levels e = {1,2} in €/s
p	penalty costs for wasted products (in €/unit)
m	fixed maximal shelf life of the product (at this age it is wasted)
h_i	holding costs/unit at each i $\in V$ per time period t $\in T \backslash \{0\}$
g_i	number of delivery days per planning period of customer i $\in V_C$
a_{ij}	distance between i & j for each i,j $\in V$
f_{ij}	average vehicle speed in m/s on each edge $i, j \in V_0 \cup V_{\mathrm{DC}}, \quad i \neq j$
l	fuel price/litre

Table 4.1: overview of all sets and parameters used in the mathematical model

4.2 Decision Variables

Variables	
$y_i^{k,t}$	1 if vertex i $\in V_{DC}$ is present on route of vehicle k $\in K_1$/ vertex i $\in V_C$ is present on route of vehicle k $\in K_2$ in period t $\in T$
$x_{i,j}^{k,t}$	1 if j is visited immediately after i by vehicle k $\in K$ in t $\in T$
$pos_i^{k,t}$	position of vertex i on route of k $\in K$ in t $\in T$
$I_i^{t,g}$	inventory of age g $\in G$ at i $\in V_0 \cup V_{DC}$ at the end of period t $\in T \backslash \{0\}$
$q_i^{k,t,g}$	quantity of age g $\in G$ shipped to i $\in V_{DC} \cup V_C$ by vehicle k in t $\in T$
W_i^t	units of waste at node i $\in V_0 \cup V_{\mathrm{DC}}$ at the end of period t $\in T$
r_t	order quantity at terminal in period t $\in T$
$\mathrm{z}_{i,t}$	1 if t $\in T$ is a delivery day for customer i in V_{DC} , 0 otherwise

Table 4.2: overview of all decision variables of the mathematical model

4.3 Complete Formulation

The problem is formulated as a mixed integer linear program (MILP).
Period 0 is modeled as a dummy period for initializing the starting inventory. In general, inventory is defined as the storage quantity that remains at a vertex i $\in V_0 \cup V_{DC}$ at the end of a period after all transportation and demand satisfaction. Like this the stock level at the beginning of a new period is the inventory of the previous day, aged by one. Products with an age of m-1 are classified as waste immediately at the beginning of the next period in order to avoid that those goods are still delivered to final customers. Since period 0 is a dummy period, no supply and ordering take place here, but operations only start in period 1 of the planning horizon.

Since the here presented formulation is an inventory routing problem, transportation as well as inventory decisions are included. The perishable goods generally arrive to the major terminal with an age of 0. The supply quantity, so the harvested amount, is assumed to be fixed and given, the order quantity however is a decision variable and will be determined by the model. The fact that the demand quantity and the number of deliveries to each customers per planning period are fixed might suggest, that the order quantity is simply the difference between demand and supply for each day. However, even though stock-outs are not allowed, this is not the case since some supply will be wasted depending on the decision about when certain customers are delivered. This decision, as chapter 6 will show, highly depends on the cost settings.

4.3.1 Objective Function

The objective is to minimize total costs, which consist of 6 factors.

(1) is the fuel consumption depending on the average speed and the travel time. The chosen fuel costs are based on the S_FUEL model in Ehmke et al. (2018), which usually

leads to an underestimation of the true fuel consumption. It only requires the gross weight of the empty delivery vehicle, which means that it is independent of the changing load. Furthermore, this model works with average speed values for each arc. This simplifies the calculation considerably and is especially applicable for food delivery, that often occurs at night. Table 4.3 summarizes the various factors that are included in this consumption model, which were also chosen as in Ehmke et al. (2018). The factor y in the objective function summarizes kN_ev.

$$\text{Minimise} \sum_{i,j \in V_0 \cup V_{\text{DC}}, i \neq j} \sum_{k \in K_1} \sum_{t \in T} \lambda (y \left(\frac{a_{i,j}}{f_{i,j}} \right) x_{i,j}^{k,t} + \gamma \beta a_{i,j} f_{i,j}^2 x_{i,j}^{k,t} + \gamma \alpha \mu x_{i,j}^{k,t} a_{i,j}) \tag{1}$$

$$+ \sum_{e \in \{1,2\}} \sum_{i,j \in V, i \neq j} \sum_{k \in K_e} \sum_{t \in T} \left(\frac{a_{i,j}}{f_{i,j}} \right) x_{i,j}^{k,t} o_e \tag{2}$$

$$+ \sum_{t \in T} r_t c_t \tag{3}$$

$$+ \sum_{g \in G} \sum_{t \in T} \sum_{i \in V_0 \cup V_{\text{DC}}} h_i^g I_i^{t,g} \tag{4}$$

$$+ \sum_{t \in T} \sum_{i \in V_0 \cup V_{\text{DC}}} W_i^t p \tag{5}$$

$$+ \sum_{i \in V_0 \cup V_{DC}} I_i^{T,m-1} p \tag{6}$$

Notation	Description	Value
λ	product of different factors from Franceschetti et al. (2013)	1/32428
k	engine friction factor	0.2
N_e	engine speed	33
v	engine displacement	5
γ	product of different factors from Franceschetti et al. (2013)	0.0028
β	product of different factors from Franceschetti et al. (2013)	1.6487
α	product of different factors from Franceschetti et al. (2013)	0.0981
μ	curb-weight	12700
l	fuel price per litre	1.8

Table 4.3: overview of the factors in the consumption model

Part (2) of the objective function models the time-dependent driver costs on both echelons while (3) are the order costs for externally procured products. Since orders are made at the major terminal and arrive only there, the order variable has only the time

period t as an index. The inventory holding costs for the major terminal and the DCs are defined in (4) and the waste penalties, which occur at the place of storage of the good at the moment of deterioration, in (5) and (6). Part (6) becomes necessary because at the beginning of each period goods are wasted if they had an age of m-1 at the end of the previous day. Like this, waste occurs immediately at the beginning of a new period and no deteriorated goods are transported and sold anymore. However, considering only (5) would ignore the waste that inevitably occurs on day T+1 based on this waste definition.

4.3.2 Model Constraints

While some constraints are added in order to define additional consideration of the newly generated problem, others, especially the routing constraints, are adaptations of constraints stemming from the formulation of Farias et al. (2020).

$$I_0^{t,0} = r_t + s_t - \sum_{k \in K_1} \sum_{i \in V_{DC}} q_i^{k,t,0} \qquad \forall\ t \in T \backslash \{0\} \tag{4.1}$$

$$I_u^{t,0} = \sum_{k \in K_1} q_u^{k,t,0} - \sum_{k \in K_2} \sum_{i \in V_{C,u}} q_i^{k,t,0} \qquad \forall\ u \in V_{DC},\ t \in T \backslash \{0\} \tag{4.2}$$

$$I_0^{t,g} = I_0^{t-1,g-1} - \sum_{k \in K_1} \sum_{i \in V_{DC}} q_i^{k,t,g} \qquad \forall\ g \in G \backslash \{0\}, t \in T \backslash \{0\} \tag{4.3}$$

$$I_u^{t,g} = I_u^{t-1,g-1} + \sum_{k \in K_1} q_u^{k,t,g} - \sum_{k \in K_2} \sum_{i \in V_{C,u}} q_i^{k,t,g} \qquad \forall\ g \in G \backslash \{0\}, u \in V_{DC},\ t \in T \backslash \{0\} \tag{4.4}$$

$$\sum_{k \in K_1} \sum_{g \in G} q_i^{k,t,g} \leq C_i - \sum_{g \in G \backslash \{0\}} I_i^{t-1,g} \qquad \forall\ i \in V_{DC},\ t \in T \tag{4.5}$$

(4.1) is the quantity of inventory of age 0 that is stored in the major terminal on each day of the planning period. It consists of the quantity which is outsourced together with the supply quantity minus what is delivered to all DCs of this age. (4.2) defines that the inventory of age 0 that is stored in DCs can only be the quantity delivered from the major terminal on the same day minus what is directly passed to the final customers on the same day. (4.3) and (4.4) deal with the inventory quantities of all other age groups, except for the deterioration age. (4.5) makes sure that the storage limits of the DCs are not succeeded.

$$\sum_{g \in G} \sum_{k \in K_2} q_i^{k,t,g} = z_{i,t} d_i \qquad \forall\ i \in V_C,\ t \in D_i \tag{4.6}$$

$$\sum_{t \in D_i} z_{i,t} = g_i \qquad \forall\ i \in V_C \tag{4.7}$$

Constraints (4.6) and (4.7) define the delivery patterns. More precisely, (4.6) only allows for delivery on days in the set D_i, which is the set of delivery days that are acceptable for

customer $i \in V_C$. (4.7), on the other hand, makes sure that the right number of delivery days is chosen, in this case it means that each customer is delivered as often as requested a week. The constant g assigned to each customer $i \in V_C$ here represents the number of times a customer wants to be visited in the planning horizon.

$$W_i^t = I_i^{t-1,m-1} \qquad \forall\ i \in V_{DC} \cup \{0\},\ t \in T\backslash\{0\} \tag{4.8}$$

$$\sum_{g \in G} I_i^{T,g} \geq \sum_{g \in G} I_i^{0,g} \qquad \forall\ i \in V_{DC} \cup \{0\} \tag{4.9}$$

$$\sum_{g \in G} q_i^{k,t,g} \leq min\left\{R_1,\ C_i\right\} y_i^{k,t} \qquad \forall\ i \in V_{DC},\ k \in K_1,\ t \in T \tag{4.10}$$

$$\sum_{g \in G} q_i^{k,t,g} \leq R_2 y_i^{k,t} \qquad \forall\ u \in V_{DC}, i \in V_{C,u},\ k \in K_2,\ t \in T \tag{4.11}$$

$$r_t \leq P \qquad \forall\ t \in T\backslash\{0\} \tag{4.12}$$

$$\sum_{u \in V_{DC}} \sum_{g \in G} q_u^{k,t,g} \leq R_1 y_0^{k,t} \qquad \forall\ \ k \in K_1,\ t \in T \tag{4.13}$$

$$\sum_{i \in V_{C,u}} \sum_{g \in G} q_i^{k,t,g} \leq R_2 y_u^{k,t} \qquad \forall\ u \in V_{DC},\ k \in K_2,\ t \in T \tag{4.14}$$

$$pos_i^{k,t} + 1 \leq pos_j^{k,t} + (|V_{DC}| + 1)(1 - x_{i,j}^{k,t}) \qquad \forall\ k \in K_1, i \in V_{DC} \cup \{0\}, j \in V_{DC}, t \in T \tag{4.15}$$

$$pos_i^{k,t} + 1 \leq pos_j^{k,t} + (|V_{C,u}| + 1)(1 - x_{i,j}^{k,t}) \qquad \forall\ k \in K_2, u \in V_{DC}, i \in V_{C,u} \cup \{u\}, j \in V_{C,u}, t \in T \tag{4.16}$$

The next constraint, (4.8), defines the waste quantity at vertex $i \in V_{DC} \cup \{0\}$ in period $t \in T$ as the inventory quantity of age m-1 at this vertex at the end of the previous day. (4.9) is an optional constraint that makes sure that the inventory quantity at each vertex at the end of the planning period is at least as high as the starting inventory. This can be required especially since otherwise the following planning period might start with very limited or no supply. (4.10) and (4.11) connect the routing variables to those for the delivery quantity. To be precise, (4.10) ensures that the quantity delivered to a DC from the major terminal by a vehicle k is only larger than 0 if the DC lies on the route of k. If this is the case, this quantity must be smaller than the minimum of the DC storage capacity and the vehicle capacity. (4.11) guarantees the same thing on the second echelon but without a capacity restriction, since final customers do not have limited storage capacity but just get their requested demand. The next constraint, (4.12), constrains the maximal order quantity to a value P, that might stand for the maximum amount that can be outsourced on each day of the week. (4.13) and (4.14) are capacity constraints for the vehicles on both echelons making sure that the total amount shipped by a specific vehicle on a specific day does not exceed the maximum vehicle load. (4.15) and (4.16) are standard sub-tour elimination constraints that resort to auxiliary variables

pos, representing the position of a vertex on the tour of a specific vehicle. The constraints guarantee for both levels, that a node j that is visited after a node i must have a position indicator that is higher of 1 than the one of i.

$$\sum_{v \in V_0 \cup V_{DC}, v \neq u} x_{u,v}^{k,t} + \sum_{v \in V_0 \cup V_{DC}, v \neq u} x_{v,u}^{k,t} = 2y_u^{k,t} \quad \forall\ u \in V_{DC},\ k \in K_1,\ t \in T \tag{4.17}$$

$$\sum_{v \in V_{DC}, v \neq u} x_{u,v}^{k,t} = \sum_{v \in V_0 \cup V_{DC}, v \neq u} x_{v,u}^{k,t} \quad \forall\ u \in V_{DC},\ k \in K_1,\ t \in T \tag{4.18}$$

$$\sum_{j \in V_{C,u} \cup \{u\}, j \neq i} x_{i,j}^{k,t} + \sum_{j \in V_{C,u \cup \{u\}}, j \neq i} x_{j,i}^{k,t} = 2y_i^{k,t} \quad \forall\ u \in V_{DC},\ i \in V_{C,u},\ k \in K_2,\ t \in T \tag{4.19}$$

$$\sum_{j \in V_{C,u} \cup \{u\}, j \neq i} x_{i,j}^{k,t} = \sum_{j \in V_{C,u \cup \{u\}}, j \neq i} x_{j,i}^{k,t} \quad \forall\ u \in V_{DC},\ i \in V_{C,u},\ k \in K_2,\ t \in T \tag{4.20}$$

$$y_0^{k,t} \leq \sum_{j \in V_{DC}} x_{0,j}^{k,t} \quad \forall\ k \in K_1,\ t \in T \tag{4.21}$$

$$y_u^{k,t} \leq \sum_{j \in V_{C,u}} x_{u,j}^{k,t} \quad \forall\ u \in V_{DC},\ k \in K_2,\ t \in T \tag{4.22}$$

$$y_i^{k,t} \leq \sum_{j \in V_{C,u} \cup \{u\}} x_{i,j}^{k,t} \quad \forall\ u \in V_{DC},\ i \in\ V_{C,u},\ k \in K_2,\ t \in T \tag{4.23}$$

$$\sum_{k \in K_e} y_i^{k,t} \leq 1 \quad \forall \begin{cases} e = 1,\ i \in V_{DC},\ t \in T \\ e = 2,\ u \in V_{DC},\ i \in V_{C,u},\ t \in T \end{cases} \tag{4.24}$$

$$\sum_{k \in K_1} y_0^{k,t} \leq b_1 \quad \forall\ t \in T \tag{4.25}$$

$$\sum_{u \in V_{DC}} \sum_{k \in K_2} y_u^{k,t} \leq b_2 \quad \forall\ t \in T \tag{4.26}$$

$$\sum_{j \in V_{DC}} x_{0,j}^{k,t} \leq 1 \quad \forall\ k \in\ K_1,\ t \in T \tag{4.27}$$

$$\sum_{j \in V \setminus \{0\}} x_{u,j}^{k,t} \leq 1 \quad \forall\ u \in\ V_{DC}, k \in K_2,\ t \in T \tag{4.28}$$

Constraints (4.17) - (4.20) are additional routing and subtour elimination constraints for both echelons and are based on the formulation of Farias et al. (2020). Their aim is to connect decision variables x and y and guarantee the flow conservation. (4.20) - (4.23) guarantee that the y variable of a vertex only turns to 1 if it is left by at least one vehicle $k \in K$. (4.24) defines that each DC as well as each customer is not visited more than once per time period. Additionally, (4.25) - (4.26) make sure that there are not more

tours performed than vehicles available in the major terminal and each DC and (4.27) - (4.28) ensure that each vehicle leaves its origin only once per time period.

$$x_{i,j}^{k,t} \in \{0,1\} \qquad \forall\ i,j \in V,\ k \in K,\ t \in T \tag{4.29}$$

$$y_{i}^{k,t} \in \{0,1\} \qquad \forall\ i \in V,\ k \in K,\ t \in T \tag{4.30}$$

$$z_{i}^{t} \in \{0,1\} \qquad \forall\ i \in V_C,\ t \in T \tag{4.31}$$

$$q_{i}^{k,t,g} \geq 0 \qquad \forall\ i \in V,\ k \in K,\ t \in T,\ g \in G \tag{4.32}$$

$$W_{i}^{t} \geq 0 \qquad \forall\ i \in V_0 \cup V_{DC},\ t \in T \tag{4.33}$$

$$r_t \geq 0 \qquad \forall\ t \in T \tag{4.34}$$

$$I_{i}^{t,g} \geq 0 \qquad \forall\ i \in V_0 \cup V_{DC},\ t \in T,\ g \in G \tag{4.35}$$

$$pos_{i}^{k,t} \geq 0 \qquad \forall\ i \in V_0 \cup V_{DC},\ t \in T,\ k \in K \tag{4.36}$$

Finally, (4.29) - (4.31) ensure that some decision variables are binary and constraints (4.32) - (4.36) define the non-negativity of the remaining decision variables. The non-negativity of the inventory that is defined in (4.35) guarantees, that no stock-outs can occur.

5 Solution Method

The problem can be solved optimally only for very small instances. For larger instances a matheuristic is proposed in this master thesis, consisting of a reduced version of the mathematical formulation presented in chapter 4 and two adaptive large neighborhood search (ALNS) procedures. First of all, the reduced mathematical model solves the first level to optimality and determines delivery patterns of final customers as well. These starting delivery patterns are optimal for the first level but they are not guaranteed to be cost-minimizing for the second level routes as well.

In the next step, second level routes are initialized by a simple cheapest insertion algorithm which is improved by a move operator. This starting solution is then further improved by a simple ALNS until a certain stopping criterion, often a fixed number of iterations, is reached. Since this first ALNS changes the second level routes by removing some customers from their current route and inserting them in a different one, in the following we will refer to this as ALNS to change 2nd level routes (ALNS-2LR). A similar combination of mathematical model and ALNS is also presented in Rohmer et al. (2019). However, while the algorithm of Rohmer et al. ends at this point, here a feedback loop is performed, again resorting to an ALNS. It works by destroying the delivery patterns of customers, which will increase the first level costs but, at best will decrease the second level routing costs enough to improve the overall solution. We will refer to this as ALNS to change delivery patterns (ALNS-DP). The following overview shows the four main steps of the proposed matheuristic.

- **Step 1:** solve a reduced mathematical model and generate:
  - ⋆ routing between terminal and DCs
  - ⋆ optimal delivery quantity to each DC on each day
  - ⋆ optimal order quantity at terminal on each day
  - ⋆ optimal delivery days for all customers (taking only first level cost factors into consideration)
- **Step 2:** based on the delivery patterns from Step 1, apply a cheapest insertion algorithm to generate routes of each DC to its customers on each day (solution of |DC| x |T|-1 independent CVRPs)
- **Step 3:** improve second level routing by ALNS-2LR
- **Step 4:** apply ALNS-DP to change delivery patterns (feedback loop)

The single steps are described in more detail in the following sections.

5.1 Reduced Mathematical Model

The reduced mathematical model is used to solve the problem without determining the second level routes. It finds the optimal inventory quantity at each DC, the order quantities, the routing between terminal and DCs and the optimal delivery days of all customers. However, the second level routing is not solved by it, which reduces the runtime and makes the problem optimally solvable even for relatively large instances. This reduction of the model is mainly done by eliminating the index k, representing the vehicle transporting a certain good, from the second level. If the first echelon is modeled as a TSP, so it is assumed that only a single vehicle delivers goods, the problem size reduces even more. Since this is the case that will be elaborated in the computational study, the following formulation is reduced by the index k on both levels. Additionally, many of the second level routing constrains become obsolete since only the quantities delivered to each customer are determined but not the precise routes that are taken to satisfy their demands.

The reduced objective function looks very similar to the original one but now (2) considers driver wages on the first level only. Since fuel consumption (1), ordering (3), inventory (4) and waste (5) and (6) affect only the terminal and the DCs, nothing changes there:

$$\text{Minimise} \sum_{i,j \in V_0 \cup V_{\mathrm{DC}}, i \neq j} \sum_{t \in T} \lambda (y \left(\frac{a_{i,j}}{f_{i,j}} \right) x_{i,j}^t + \gamma \beta a_{i,j} f_{i,j}^2 x_{i,j}^t + \gamma \alpha \mu x_{i,j}^t a_{i,j}) \tag{1}$$

$$+ \sum_{i,j \in V_0 \cup V_{DC}, i \neq j} \sum_{t \in T} \left(\frac{a_{i,j}}{f_{i,j}} \right) x_{i,j}^t o_1 \tag{2}$$

$$+ \sum_{t \in T} r_t c_t \tag{3}$$

$$+ \sum_{g \in G} \sum_{t \in T} \sum_{i \in V_0 \cup V_{\mathrm{DC}}} h_i^g I_i^{t,g} \tag{4}$$

$$+ \sum_{t \in T} \sum_{i \in V_0 \cup V_{\mathrm{DC}}} W_i^t p \tag{5}$$

$$+ \sum_{i \in V_0 \cup V_{DC}} I_i^{T,m-1} p \tag{6}$$

As already mentioned, all decision variables which previously included the vehicle k are now reduced by this index. Furthermore, some second level routing constraints become obsolete. The new constraints look as follows (please note that non-negativity and integrality-constraints are omitted here, since they are equivalent to the ones from the complete formulation in 4):

Subject to

$$I_0^{t,0} = r_t + s_t - \sum_{i \in V_{DC}} q_i^{t,0} \qquad \forall\ t \in T\backslash\{0\} \tag{5.1}$$

$$I_u^{t,0} = q_u^{t,0} - \sum_{i \in V_{C,u}} q_i^{t,0} \qquad \forall\ u \in V_{DC},\ t \in T\backslash\{0\} \tag{5.2}$$

$$I_0^{t,g} = I_0^{t-1,g-1} - \sum_{i \in V_{DC}} q_i^{t,g} \qquad \forall\ g \in G\backslash\{0\}, t \in T\backslash\{0\} \tag{5.3}$$

$$I_u^{t,g} = I_u^{t-1,g-1} + q_u^{t,g} - \sum_{i \in V_{C,u}} q_i^{t,g} \qquad \forall\ g \in G\backslash\{0\}, u \in V_{DC},\ t \in T\backslash\{0\} \tag{5.4}$$

$$\sum_{g \in G} q_i^{t,g} \leq C_i - \sum_{g \in G\backslash\{0\}} I_i^{t-1,g} \qquad \forall\ i \in V_{DC},\ t \in T \tag{5.5}$$

$$\sum_{g \in G} q_i^{t,g} = z_{i,t} d_i \qquad \forall\ i \in V_C,\ t \in D_i \tag{5.6}$$

$$\sum_{t \in D_i} z_{i,t} = g_i \qquad \forall\ i \in V_C \tag{5.7}$$

$$W_i^t = I_i^{t-1,m-1} \qquad \forall\ i \in V_{DC} \cup \{0\},\ t \in T\backslash\{0\} \tag{5.8}$$

$$\sum_{g \in G} I_i^{T,g} \geq \sum_{g \in G} I_i^{0,g} \qquad \forall\ i \in V_{DC} \cup \{0\} \tag{5.9}$$

$$\sum_{g \in G} q_i^{t,g} \leq min\left\{R_1,\ C_i\right\} y_i^t \qquad \forall\ i \in V_{DC},\ t \in T \tag{5.10}$$

$$r_t \leq P \qquad \forall\ t \in T\backslash\{0\} \tag{5.11}$$

$$\sum_{u \in V_{DC}} \sum_{g \in G} q_u^{t,g} \leq R_1 y_0^t \qquad \forall\ t \in T \tag{5.12}$$

$$\sum_{i \in V_{C,u}} \sum_{g \in G} q_i^{t,g} \leq R_2 y_u^t \qquad \forall\ u \in V_{DC},\ t \in T \tag{5.13}$$

$$pos_i^t + 1 \leq pos_j^t + (a+1)(1 - x_{i,j}^t) \qquad \forall\ i \in V_{DC} \cup \{0\}, j \in V_{DC}, i \neq\ j, t \in T \tag{5.14}$$

$$\sum_{v\ \in V_0 \cup V_{DC}, v \neq u} x_{u,v}^t + \sum_{v \in V_0 \cup V_{DC}, v \neq u} x_{v,u}^t = 2y_u^t \qquad \forall\ u \in V_{DC},\ t \in T \tag{5.15}$$

$$\sum_{v\ \in V_{DC}, v \neq u} x_{u,v}^t = \sum_{v \in V_0 \cup V_{DC}, v \neq u} x_{v,u}^t \qquad \forall\ u \in V_{DC},\ t \in T \tag{5.16}$$

$$y_0^t \leq \sum_{j \in V_{DC}} x_{0,j}^t \qquad \forall\ t \in T \tag{5.17}$$

$$y_0^t \leq 1 \qquad \forall\ t \in T \tag{5.18}$$

$$\sum_{j \in V_{DC}} x_{0,j}^t \leq 1 \qquad \forall\ t \in T \tag{5.19}$$

5.2 Generation of the second level routes

After the order and waste quantities, inventory levels and routes on the first level are, together with the delivery days of final customers, determined by the reduced mathematical model, the next step is to generate the routes on the second echelon. First, a set of starting routes is obtained by a simple cheapest insertion. Those routes are, in the next step, improved by the ALNS-2LR. The implemented algorithm is an adaptation of the ALNS presented in Hemmelmayr et al. (2012), which extends the original ALNS from the work of Pisinger and Ropke (2006) to a two echelon case. However, the ALNS of Hemmelmayr et al. (2012) does not include inventory or waste-related considerations.

After generating an initial routing solution for each DC on each day of the planning period, these routes are first destroyed to a certain extent by a set of destroy operators. The used destroys operators take some customers out of the current incumbent solution and put them into a customer pool. Afterwards, in the repair step, they are reinserted into the solution based on different considerations. Promising solutions are improved additionally by a classical move operator. Destroy and repair operators are selected by a roulette wheel selection based on their scores, which increase every time they find an improvement. Like this, operators which performed well in previous iterations are more likely to be chosen again. The following pseudo-algorithm, that is inspired by Hemmelmayr et al. (2012), summarizes the approach:

Algorithm 1 ALNS-2LR

$s \leftarrow InitialSolution, InitializeScores(\pi)$
while Stopping Criterion Not Reached **do**
  $N_D \leftarrow ChooseDestroyOperator(D, \pi)$
  $N_R \leftarrow ChooseRepairOperator(R, \pi)$
  $s' \leftarrow DestroyAndRepair(s, N_D, N_R)$
  if $f(s') < (1 + a)f(s*)$ **then**
    $s' \leftarrow ImproveWithMoveOperator(s')$
  end if
  if $f(s') < f(s)$ **then**
    $s \leftarrow s'$
  end if
  if $f(s) < f(s*)$ **then**
    $s* \leftarrow s$
    $UpdateScores(\pi)$
  end if
end while

D is the set of destroy operators and R the set of repair operators. In the computational study, a fixed number of iterations is used as a stopping criterion. After this criterion is met, the algorithm returns the best-found solution s* with its costs and the time needed for running the whole algorithm. In the next section the single elements of the algorithm are described in more detail.

5.2.1 Search Space

Some repair operators, namely all those which insert customers individually, check for feasibility of the chosen insertion position and allow only feasible insertions. However, since there is a set of operators that insert the whole customer pool, or part of it, as one, those can insert sequences of customers in any position, no matter if the capacity constraints hold or not. The resulting tours are almost always infeasible. If the total costs look promising, i.e. they lie within a certain range of the best solution found so far, a split operator is applied to make them feasible. This works by turning them into one single giant tour and splitting them up in the cheapest, feasible way.

5.2.2 Cheapest insertion to generate the initial solution

As already mentioned, the assignment of customers to DCs is fixed and given, which is realistic in a case with cargo bikes with limited reach and loading capacity. For this reason, after the reduced mathematical model is solved and the starting delivery patterns are determined, the remaining problem is a simple Capacitated VRP (CVRP) for each DC on each day of the planning period. Like this the problem that is solved by the ALNS-2LR procedure is a set of |DC| x |T|-1 CVRPs, where |DC| is the number of distribution centres and |T|-1 the number of periods in which customers are delivered.

For each of these CVRPs, the initial solution is generated by randomly sorting the customers, that are delivered on this day. All customers are inserted, one after the other, into their cheapest position of the current tour. If a customer does not fit into any of the existing tours, a new one is opened. This simple starting solution can, if necessary, be improved by the move operator, which will be described in more detail in 5.2.5. This improvement step is used for all instances in the computational study, see chapter 6.

5.2.3 Destroy Operators

The destroy operators which are used in the ALNS-2LR all get the number of customers to delete as an input. They all work similarly and differ mainly in the logic they apply to select the customers that are removed from the current incumbent.

Random removal

Random removal is a standard operator that was already used by Pisinger and Ropke (2006) in the first implementation of the ALNS. The presented operator selects a random tour from the incumbent solution and puts a random customer from this tour into the customer pool. This is repeated until the desired number of customers is removed from the solution.

Random tour removal

This is an operator equal to the route removal operator from Hemmelmayr et al. (2012) and functions very similarly to random removal. The only difference is that here the

input is the number of tours to be deleted. This number of tours is selected randomly, and all customers currently delivered by this tour are put in the customer pool, leaving the tour empty.

Worst removal

The worst removal operator removes a fixed number of customers with the highest removal gain. As in Pisinger and Ropke (2006) and in Hemmelmayr et al. (2012) the removal gain is calculated as the cost reduction achieved if a customer is taken out of its current tour and put into the customer pool, so the travel costs from its predecessor to the customer and from the customer to its successor.

Worst removal from each tour

Similar to worst removal, this operator also deletes a given number x of customers with the highest removal gain, but this time from each tour individually. The algorithm does not remove anything from tours that have less than x customers in them already.

Remove related

This operator gets two inputs: the number of customers to remove, x, and a randomly chosen seed customer. It then deletes the x customers with the lowest travel costs to the seed customer from the current incumbent and puts them into the customer pool.

5.2.4 Repair Operators

All the repair operators in the present algorithm do not take any input, but they simply work on the customer pool that was filled by the destroy operators. Some of them insert customers individually whereas others insert the complete customer pool, or a sequence of customers in it, into the solution. As already mentioned, all operators which insert individual customers only allow for feasible insertions. If a sequence of customers is inserted, the split operator, that is described in 5.2.5 in more detail, is used to make the resulting tours feasible.

Greedy insertion each single customer (with and without perturbation)

The applied greedy insertion operators are inspired by the implementation of Pisinger and Ropke, who use insertion costs to determine the best insertion position for each customer. These insertion costs are calculated as the cost difference of the solution without the customer compared to its costs with the customer inserted at a specific position.
In more detail, the operator works as follows: The first customer from the customer pool is chosen and its insertion costs for each possible insertion position are calculated. Then it is checked, if inserting the customer in the cheapest possible position leads to a feasible solution. If not, the next cheapest one is tested and so on. If no insertion position leads to a feasible solution a new tour with just this customer in it is opened. Then, the whole

procedure is repeated for the next customer until the whole pool is inserted.
In the version with perturbation only the calculation of the insertion costs changes. Here, after calculating the insertion costs of a customer in all possible positions, these costs are multiplied with a random value between 0.8 and 1.2. This step stems from Hemmelmayr et al. (2012) and integrates some randomization into the procedure.

Greedy insertion whole pool (with and without perturbation)

In this version of greedy insertion the whole customer pool is randomly sorted at the beginning. The cost of inserting the whole pool in this order is then calculated for each possible insertion position. It is now inserted in the cheapest possible position, no matter if this is feasible or not. Finding a feasible insertion position for the whole customer pool is usually impossible except for a case where only very few customers are in the pool. The infeasible solution is then made feasible by a split procedure and only considered further if its costs are within a certain range of the current best solution. Otherwise, the algorithm stops here and the previous solution undergoes a destroy and repair phase again.
As in the previous one, also for this operator a version with perturbed insertion costs was implemented. The factor used for perturbing the insertion costs is the same as in the previous operator.

Greedy insertion forbidden

The greedy insertion forbidden operator uses the same insertion criterion as the previous ones. The only difference is that it is not allowed to include a customer in the same tour it was previously assigned to. This means that each customer is first tried to be inserted in the best insertion position. If this is feasible it is also checked, if this is the tour that the customer was part of before the destroy operator. If this is the case, the next best insertion position is evaluated until one in a different tour that is feasible is found. However, in case no such insertion position exists, the customer is inserted in an empty tour.

Regret insertion

In the regret insertion operator the regret value of inserting a customer is calculated for each customer in the customer pool. The regret value here is not simply defined as the difference between the best and the second-best insertion position but as the sum of the differences between the best insertion position and any other. This regret-k-heuristic is considered to take future iterations more into account as a normal regret heuristic. The customers are then inserted, using the greedy insertion each single customer operator, in decreasing order of their regret values. As all operators that insert each customer individually also here only feasible solutions are produced.

5.2.5 Additional Operators and Local Search

There are two additional operators that are used in the procedure. The split is used to generate feasible solutions if the whole customer pool was inserted at once, which almost always leads to an infeasible tour. The move is used on promising solutions to improve them additionally in a simple Local Search.

Split

The aim of the split operator is to make sure that all tours are feasible, hence that the total demand of each tour does not exceed the capacity constraint of the vehicles. This operator merges all the tours of the incumbent solution into a giant tour. In the next step customers are, as long as the sum of their demand does not exceed the capacity limit, put into the first tour. The order is not changed in this procedure. Once the capacity limit is reached, a new tour is opened and the next customers are added. This procedure continues until all customers are included in feasible tours.

Local Search

The move operator is used to improve promising solutions additionally if possible. To do so, it goes through all customers in the order they have in the current solution and calculates their insertion costs in any other position. The list of insertion costs is then sorted in increasing order. In the next step, the insertion costs are compared to the gain of removing the customer from its current position. As long as the removal gain is higher than the insertion costs the algorithm checks if inserting a customer in this position is feasible. If no position that would lead to an improvement allows for a feasible insertion of the customer, it stays in its current position. Otherwise, the best feasible insertion position is chosen, and the customer moves to this new position. Like this only improvements and only feasible new solutions are generated.

5.3 Complete ALNS-2LR procedure

As soon as the input data is defined, the initial solution is produced by the cheapest insertion algorithm as described in section 5.2.2 and improved by the move operator. Tours are represented by a dictionary with the tour number as a key (starting with tour number 1) and the list of customers in the order of their visit as value.

The scores of all destroy and repair operators are initialized as 1. These scores are then transformed into probabilities by summing up all scores for one type of operator and dividing each individual score by this sum. They are stored in a cumulative manner, such that, for example the probabilities of the 5 destroy operators at the beginning are $\frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}$ and 1. Like that, all operators have the same chance of being chosen at the beginning. The roulette wheel selection is then implemented by generating a random number between zero and one. If this number lies between the probability of one operator

and another, the second one of them is chosen. A fixed number of iterations is used as a stopping criterion in order to make the results for different data sets easy to compare.

The customer pool is initialized as empty and the roulette wheel selection functions are called to choose a destroy and a repair operator randomly. The chosen destroy operator is then applied on the starting solution and takes some customers out of this solution to put them into the customer pool. After this, the repair operator is used to reinsert the customers based on its individual insertion logic. After this, the total costs (or distance) of the previous and this newly created solution are compared. As already mentioned, all repair operators that insert single customers one at a time produce feasible solutions already. If the whole customer pool was inserted at once instead, at this point it is checked if the new solution is not more than a given percentage worse then the previous one. If the solution is already very costly even though it is not even feasible, the iteration is interrupted and the procedure starts again. Otherwise the split operator is used to make it feasible. After this step, the new, feasible solution is checked for its quality again. This time, its total distance is compared to the best solution found so far. If it is not more than a fixed percentage worse than the best found solution, the move operator is applied to improve the solution further. Otherwise, it is discarded and the procedure is again interrupted and a new iteration starts. At this point it is checked whether the current incumbent is better than the previous incumbent solution. If yes, it is used as a new starting solution and the procedure starts again. If not, the previous solution is used as a starting solution again in the next iteration. The last step for each iteration is saving the solution as the current best if its total costs are lower than in every previously found one. If a new best solution was found in an iteration, the score of the used destroy and repair operator is increased by one, which increases their chance of being chosen again in the following iterations.

After the stopping criterion is reached, the code prints the tours and the total distance of the best found solution, the number of iterations that were performed and a list of all iterations, in which an improvement was found. It also prints the final scores of each destroy and repair operator, which give some information about how successful each of them was. Since in the computational experiments it is assessed, if there are certain combination of destroy and repair operators, that are especially successful, the code also counts how often each specific combination was successful. An overview of this analysis on some known instances will be given in chapter 6.

5.4 ALNS-DP and feedback loop

The combination of the reduced mathematical model and the ALNS-2LR procedure leads to a complete solution to the problem. However, like this the two echelons are optimized mostly independently. Even though the solution is optimal for the first echelon, including routing and inventory decisions, it might be possible to change the delivery patterns of final customers and decrease the second echelon routing costs by doing so. Changing delivery patterns, again impacts the inventory levels and routes on the first echelon. Therefore, this is only useful in a case where the cost reduction on the second echelon is

higher than the cost increase on the first level. In general, two echelon problems aim at solving all stages together since this is expected to lead to a better overall solution.

For this reason the present matheuristic does not stop at this point but a second, larger impact ALNS, the ALNS-DP, is applied. Its purpose is to change the days on which customers are delivered to different days within their set of acceptable delivery periods.

The ALNS-DP uses the result of the ALNS-2LR as a starting solution and applies destroy operators to remove some customers from their current delivery days. The repair operators then insert these customers into new periods out of the set of acceptable delivery days of this customer. To illustrate the general idea behind this, the example from 3.2.1 will be used. Let us assume, that the following table shows the delivery patterns that were found by the reduced mathematical model:

Customer	Day 1	Day 2	Day 3	Day 4	Day 5
A (1)		•			
B (2)				•	•
C (1)		•			
D (2)			•		•
E (1)					•

Table 5.1: Example for delivery patterns found by the reduced mathematical model for customer A to E in a planning horizon of five days. Again, grey cells are days on which a customer is unavailable and white cells stand for possible delivery days. While customers A, C and E require one visit per week, B and D must be visited twice. The black circles again represent chosen delivery days.

Let us consider a case where customers A to E are assigned to the same DC. This DC has no demand to fulfil on day 1, so no tours are performed on this day by its cargo bikes. On day 2 customers A and C are visited in a tour {DC, A, C, DC}. On both, day 3 and 4, just one customer is visited, which leads to tours {DC, D, DC} and {DC, B, DC} while on day 5 customers B, D and E are delivered. The best found tour here in the example was {DC, B, D, E, DC}. This solution can be destroyed by changing the delivery days of, say customer D from day 3 and day 5 to day 3 and day 4, as visualized in table 5.2.

Customer	Day 1	Day 2	Day 3	Day 4	Day 5
A (1)		•			
B (2)				•	•
C (1)		•			
D (2)			•	•	
E (1)					•

Table 5.2: Possible alternative delivery patterns that might be more expensive on the first echelon but cheaper on the second one.

In the algorithm such a change is performed by deleting customer D from all tours it is currently part of, so {DC, D, DC} and {DC, B, D, E, DC}. Next, customer D is reinserted based on some insertion logic, which depends on the chosen operator, in our case into the tours of day 3 and 4. If this solution leads to a decrease in transportation costs on the second echelon that is high enough to to justify the resulting cost increase on the first level, a better overall solution was found.

Again, there are different destroy and repair operators in use, which work similarly to the ones in the ALNS-2LR. However, there are some differences, which will be described in the following section.

5.4.1 Solution Representation and Operators

First of all, the solution is now not split into simple CVRPs for each DC in each period but all periods of a certain DC must be evaluated simultaneously. Therefore, the solution is represented by a nested dictionary, containing all current routes in all periods for a certain DC. Each destroy operator deletes the chosen customer from all the routes it is included in, in all periods. Afterwards, the repair operators insert the customer into different delivery patterns, which might be different or the same as before.

Even though the operators are applied to larger sets of tours, the logic used for destroying and repairing does not change. The ALNS-DP again resorts to the operators random removal, random tour removal, worst removal and remove related. The insertion operators are only applied in the version that inserts each customer individually since the insertion of a group of customers at once does not make sense in a setting where each customer has different possible delivery days. Furthermore, the set of destroy operators is reduced to greedy insertion, greedy insertion perturbation and regret insertion. All of them make sure that all capacity constraints hold. Greedy insertion forbidden is not used in the ALNS-DP since not all customers might be willing to accept delivery on at least twice as many days as they are delivered, so changing delivery patterns completely is mostly infeasible.

5.4.2 Complete Algorithm for the ALNS-DP

The ALNS-DP is constructed very similarly to the ALNS-2LR. The tours of one DC in all periods are used as an starting solution. Destroy and repair operators are again chosen by a roulette wheel selection starting with the same probability for all operators.The destroy operators remove a given number of customers from all the tours they are included in and put them into the customer pool. This customer pool is given to the repair operators as an input, which reinsert each customer in as many of its available delivery days as required. The total costs of the new solution are compared to the costs of the previous one and improved by the move operator if it is not more than a certain percentage more costly compared to its predecessor. Otherwise the new solution is discarted and the previous incumbent is destroyed and repaired again in a new iteration. If the new solution, after the move operator, is better then the previous one, it becomes the new incumbent and is

stored as best solution found so far. Whenever an improvement is found, the scores of the successful destroy and repair operator are again increased.

5.5 Description of the feedback loop

As already mentioned, the ALNS-DP is part of a feedback loop that aims at improving the solution found by the reduced mathematical formulation and the ALNS-2LR. This feedback loop starts by calling the ALNS-DP and applying it on this starting solution. The costs of the best solution found are compared to the original second level costs. After a fixed number of iterations, the best solution found is chosen and the reduced mathematical model is solved with the corresponding delivery patterns as an input. Note that the mathematical model does not change in this step, but the delivery patterns are now a given input instead of a decision to be made.

If the new total costs after the ALNS-DP and the solution of the corresponding first level are not more than some percentage a worse than before the feedback loop, the ALNS-2LR is used again on the second level to eventually detect additional improvements.

If the sum of the found first and second level costs is not more than a small percentage b worse than the initial solution, it becomes the new incumbent for the next iteration of the feedback loop. Otherwise we go back to the original solution and repeat the procedure. The following pseudo-algorithm summarizes the steps of the feedback loop:

Algorithm 2 Feedback Loop

$s2 \leftarrow StartingSolution2ndLevel$
$s1 \leftarrow StartingSolution1stLevel$
$p \leftarrow CurrentDeliveryPatterns$
while Stopping Criterion Not Reached **do**
  $(p', s2') \leftarrow ChangeDeliveryPatterns(p, s2)$
  if $f(s2') < f(s2)$ **then**
    $s1' \leftarrow Solve1stLevel(p')$
    if $f(s2') + f(s1') < (1 + a) * (f(s1) + f(s2))$ **then**
      $s2' \leftarrow ALNS - 2LR(s2')$
    end if
    if $f(s2') + f(s1') < f(s2) + f(s1)$ **then**
      $s2* \leftarrow s2'$
      $s1* \leftarrow s1'$
    end if
    if $f(s2') + f(s1') < (1 + b) * (f(s2) + f(s1))$ **then**
      $s2 \leftarrow s2'$
      $s1 \leftarrow s1'$
    end if
  end if
end while

The function ChangeDeliveryPatterns here stands for the ALNS-DP procedure, as described in 5.4.2.

This feedback loop is more computationally expensive than the first part of the algorithm, since the underlying problem is larger and more complex. For this reason, the computational study will mainly focus on testing how many iterations of the loop are usually necessary to find the first improvement for different instances. Other stopping criteria that were used are a fixed run time and a fixed number of improvements found.

6 Computational Study

The computational study is separated into two main parts. The first one examines the success of the ALNS-2LR on five known CVRP instances from the literature of different size. The computational experiments performed on these instances also define the parameters for the final version of the code. On the other hand, in the second part of the computational study the complete algorithm, including the reduced mathematical model, the ALNS-2LR and the feedback loop with the ALNS-DP, is tested on new instances. For very small instances with one DC and 10 customers the solution can be compared to the optimal found with the MILP. For larger instances just the improvements can be reported.

The newly generated instances used for testing the algorithm have different properties and are used to assess, how successful the algorithm is for different cost structures in practical problems.

All experiments were performed on a single machine with an AMD Ryzen 7 3700U processor running at 2.10 GHz. The heuristic was coded in Python 3.8.12 and the Gurobi Python interface gurobipy. The first set of experiments was run on Google Colaboratory while the second was, for licencing reasons, performed on the scientific environment Spyder IDE.

6.1 Evaluation of the ALNS-2LR

The first part of the computational study deals with the ALNS-2LR, so the one that works on the single CVRPs of each DC in each period. Since the underlying problem is relatively simple and well-known, it can be tested on instances from the literature and its performance can be evaluated in comparison to the optimal. This first set of computational experiments was performed on five test instances from Uchoa et al. (2014), which are provided online in the CVRP Library by Xavier et al. (2014). The size of the chosen instances ranges between 100 and 199 customers. For each set, the vehicle capacity and the customer demands are given.
Table 6.1 summarizes the chosen instances and the performance of the ALNS-2LR on them compared to the proven optimum.

In the table, the columns "Customers" and "Capacity" show the properties of the instance, namely their size and vehicle capacity. Next, the average solution costs over five runs with 1000 iterations of the ALNS-2LR are reported, together with the best found solution and the proven optimum, that was found by different algorithms in literature (Xavier et al., 2014). As the evaluation part of the table shows, the average runtime increases for the larger instances. Also, the percentage GAP between the best found

Data Set	Properties		ALNS-2LR Performance			Evaluation	
	Customers	Capacity	Average	Best	Optimal	time (s)	GAP
X-n101-k25	100	206	29 240	28 756	27 591	114	4.22
X-n125-k30	124	188	59 487	58 997	55 539	207	6.22
X-n153-k22	152	144	22 557	22 179	21 220	241	4.52
X-n176-k26	175	142	51 466	50 917	47 812	299	6.49
X-n200-k36	199	402	62 700	61 977	58 578	384	5.08

Table 6.1: Performance of the ALNS-2LR on 5 different instances

solution and the proven optimum is presented. This GAP is always around 5% already for the 1000 iterations performed. The algorithm performed worst for the second largest instance with 175 customers, where the best found solution was still more than 6 % worse than the proven optimum.

As already mentioned, a fixed number of 1000 iterations was used as a stopping criterion to increase the comparability of different runs. The percentage of the solution destroyed each time depends on the chosen destroy operator and is, for each operator, generated randomly in a certain range, depending on the customer number. For instance, the number of tours removed by the random tour removal operator ranges between 1 and half of all current tours.

One more thing that is studied experimentally with the previously mentioned instances is the success of the different destroy and repair operators and their combinations. Table 6.2 summarizes, how often each destroy operator was able to find an improvement in 1000 iterations performed on all five instances (please note that the instances are summarized by their customer number n).

operator	n100	n124	n152	n175	n199	total
random removal	21	8	6	8	34	77
random tour removal	16	29	5	13	41	94
remove related	1	9	0	0	4	14
worst removal	1	4	0	0	6	11
worst removal from each tour	0	4	0	0	17	21

Table 6.2: Number of improvements found by the destroy operators for 5 instances

As this overview shows, the random removal performs well on all instances and the random tour removal finds most improvements overall. However, all remaining operators did not find any improvement for some instances. The worst removal performed very poorly on all instances except for the largest one, which might imply that this is an efficient operator for larger size instances.
The same analysis was done for the repair operators and is summarized in table 6.3:

operator	n100	n124	n152	n175	n199	total
greedy insertion each single customer	8	9	2	6	24	49
greedy insertion perturbation single customer	9	19	0	2	56	81
greedy insertion whole pool	2	1	0	0	2	5
greedy insertion perturbation whole pool	2	0	2	1	1	6
greedy insertion forbidden	8	6	1	7	2	24
regret insertion	12	17	5	5	18	57

Table 6.3: Number of improvements found by the repair operators for 5 instances

As the table shows, the most successful repair operators are greedy insertion perturbation each single customer, regret insertion and greedy insertion each single customer. Greedy insertion perturbation whole pool did not improve the solution in many cases.

As already mentioned in the description of the solution method, all operators start with a score of one and have an equal chance of being chosen at the beginning. Since the algorithm rewards successful operators, every time an improvement of the solution has been found, the score of the used operators is increased by one. Increasing or decreasing this score update per iteration did not lead to any considerable differences in the final outcome in the performed computational experiments.

The next thing that was tested is, if some operators work especially well in combination. Therefore, it was examined how often certain destroy and repair operators succeed in collaboration. On the smallest instance, the most successful combinations were the following:

- ⋆ random removal & greedy insertion single customer (7 improvements found)
- ⋆ random removal & regret insertion (7 improvements found)

The solution of the instance with 125 customers was improved most times by the following combinations:

- ⋆ random tour removal & greedy insertion perturbation each single customer (10 improvements found)
- ⋆ random tour removal & greedy insertion each single customer (8 improvements found)

For the next larger instance with 150 customers the most successful combinations were:

- ⋆ random tour removal & regret insertion (3 improvements found)
- ⋆ random removal & regret insertion (2 improvements found)

For 175 customers, the following destroy and repair operators cooperated best:

- ⋆ random tour removal & greedy insertion single customer (5 improvements found)

- ⋆ random removal & greedy insertion forbidden (4 improvements found)

And, last but not least, for the largest instance with 199 customers the best combination of destroy and repair operators was:

- ⋆ random tour removal & greedy insertion perturbation single customer (26 improvements found)
- ⋆ random removal & greedy insertion perturbation single customer (13 improvements found)

No combination, however, consistently outperformed others. Therefore, the repair and destroy operators in the final version of the algorithm are still chosen separately by the roulette wheel selection, and no specific combination has a higher chance of being selected than any other.

The last thing that was assessed for the ALNS-2LR is the performance of the simple local search, consisting only of a move operator. In the algorithm, the local search is used to improve solutions that are not more than 10 % worse than the best solution found so far. To assess the gained value of the local search, the average cost improvement it caused over 100 iterations is printed by the code.

Instance	Min	Max	Avg	Avg %
X-n101-k25	425	1077	705	2.56
X-n125-k30	640	1456	877	1.58
X-n153-k22	219	433	304	1.43
X-n176-k26	204	809	588	1.23
X-n200-k36	684	1086	885	1.51

Table 6.4: Cost improvements found by the move operator over 100 runs on 5 different instances

As table 6.4 shows, for the smallest instance this average improvement ranges between 425 and 1077 cost units, the average improvement per run of the move are 705. This corresponds to 2.56 % of the proven optimum of this instance. The outcomes for the larger instances are similarly good. The conclusion is, that the move operator makes a valuable contribution to the algorithm and is therefore kept for the remainder of the computational study. Applying it to relatively many solutions, thus for all that are up to 10 % worse than the current best, also seems reasonable for the same consideration.

6.2 Evaluation of the complete matheuristic

6.2.1 Comparison to optimal solution

As already mentioned, the complete matheuristic consists of three main parts, the reduced mathematical model, the ALNS-2LR and the feedback loop with the ALNS-DP. This

section of the master thesis evaluates these three parts for newly generated instances with different properties. The first step here is generating a small, random instance with 10 customers and one DC to test if the algorithm can find the optimal solution from the MILP. The first five customers in this instance are visited once, the second halve twice in the time horizon. Each customer gets a random set of two to four possible delivery days assigned and the demand is randomly generated in a range of 1 to 50. The vehicles in the DC have a capacity limit of 100 and the supply lies between 10 and 100 units per day. For this instance with a grid size of 100 times 100, the proven optimum was found with an average run time of 65.094 seconds over five runs with 1000 iterations of the feedback loop. The fastest run took only 55.70 seconds to find the optimal solution, the worst still found it in 68.56 seconds. This proves that the matheuristic works, at least for small instances that are still optimally solvable with the complete mathematical model - which in this case took the solver gurobipy around 48 seconds.

Solving an instance with 12 customers and a capacity limit of 120 for the cargo bikes already took the solver 3329 seconds. The matheuristic solved this to optimality in three out of five runs within 1000 iterations, as table 6.5 shows:

Run	Costs without 2nd level routing					Results		
	fuel	driver	order	holding	waste	Total	time (s)	GAP
optimal	72.53	35.81	0.00	10.03	16.10	928.29	3329	
1	72.52	35.80	0.00	10.03	16.10	933.07	103	0.51
2	72.52	35.80	0.00	10.03	16.10	928.29*	64	0.00
3	72.59	35.85	0.00	10.03	16.10	928.42*	48	0.00
4	72.53	35.81	0.00	10.03	16.10	933.07	90	0.51
5	72.53	35.81	0.00	10.03	16.10	928.29*	166	0.00

Table 6.5: Performance of the complete matheuristic compared to the optimal solution found by the solver (12 customers)

The table shows the value of the five cost elements of the objective function and the total costs in the best found solution of the algorithm compared to the optimal solution found by solving the complete MILP. The matheuristic never took more then 166 seconds, in the fastest run 48 seconds were enough to find the optimal solution. In the worst case the difference to the optimum was still only 0.51 %.

Please note that the internal supply, so the quantity of the good produced by the company itself, for this instance was relatively high, which caused that no ordering was necessary. This is the most realistic assumption for our business partner since the company stated, that they usually prefer overproduction instead of not being able to satisfy all customers with the desired products.

Since the problem is np-hard, for larger instances the MILP is not optimally solvable anymore, at least not in an acceptable amount of time. Therefore, the novel matheuristic

is next tested on new instances. All instances are generated randomly but with a fixed seed in order to make the results reproducible. In all cases the DCs are supplied by a single vehicle without a strong capacity limitation. This setting guarantees that the solution of the reduced mathematical model to optimality is possible in a considerable time frame. However, it is easily extendable to a case with multiple vehicles on the first echelon by minor adaptations of the model. In this case the solution of the problem can be stopped at, depending on the size of the problem, e.g. 5 % from optimum, which again guarantees a reasonable runtime of the complete algorithm. As a result, since this computational study mainly aims at testing the performance of the algorithm on instances with different cost settings, the case with a TSP on the first echelon and multiple vehicles (CVRPs) on the second, is evaluated. The time horizon always consists of T = 5.

6.2.2 Description of the Instances

The initial random instance contains N=400 customers and dc=6 distribution centers. There is only $b_1 = 1$ vehicle available on the first echelon and $b_2 = 7$ cargo bikes in each DC. Customers are delivered in a planning period of 5 days and products deteriorate at an age of 4, which means that their third day of life is the last one on which they can be sold. The capacity of the large combustion-engine vehicle is 8500 units and 750 units for the cargo bikes. Not more than 1850 units can be stored in each DC, the terminal has unlimited storage capacity. The starting inventory of each age in each DC is generated randomly and lies between one and 200 units, the wage of drivers on the first echelon is 0.001 monetary units per m, while a speed of 9 m/s is assumed. Since cargo bikes usually move slower, drivers here get 0.003 monetary units per m and their speed is assumed to be 3 m/s on average. These values are chosen based on Soysal et al. (2015). Customers have fixed demands of 1 to 100 units and the supply in each period is randomly generated in a range of 1000 to 5000 units per day. A random set of possible visit days is assigned to each customers, which lies between 2 and 4 days. Note that, if a customer that is delivered twice a week gets only two days assigned, the delivery pattern of this customer is already fixed. Half of the customers is classified as "one visit customer", the other half is visited twice. The holding costs are dependent on the age of the product, as in Rohmer et al. (2019), in order to penalize the delivery of older products to customers by higher costs. This already contributes to waste reduction since it incentivizes the delivery of fresher products to final customers. If the remaining life time at customers is higher, they have more time to consume them and the probability that products are wasted already decreases. For this reason, even though these costs are classified as holding costs in the objective function, they could also be considered waste costs.

The development of holding costs is modeled as follows: they start with 0.005 monetary units per unit for age 0 in the DCs and increase by 0.001 for every day a product ages. In the terminal the holding costs are considered 0.007 monetary units more expensive per unit for all ages. Penalty costs for a unit of waste are set to 0.1 at the beginning and the order costs in each period per unit lie between 0.006 an 0.04 monetary units per piece. However, the cost setting will be varied in different runs in order to test how high the influence of different cost values is on the result. Table 6.6 shows the cost values from the

objective function, resulting from a solution of the reduced mathematical model with the previously described cost setting:

Cost Factor	Cost Value
fuel costs	353.00
driver costs	93.85
order costs	159.66
holding costs	48.51
waste costs	82.60

Table 6.6: Cost factor values after the solution of the first echelon with the reduced mathematical model

The total objective value of the first echelon in this case is 737.63.

Table 6.7 shows how the total costs evolve in the process of the algorithm. Please note that in this case the algorithm is stopped as soon as the feedback loop finds a first improvement. The last column in the table shows how many iterations of the feedback loop were necessary to improve the total costs for the first time.

Run	Step 1		Step 2		Comparison		
	1st E	2nd E	1st E	2nd E	Total before	Total after	iterations
1	737.63	8389.72	740.81	8313.10	9127.35	9053.91	2
2	737.63	8330.80	766.69	8301.49	9068.43	9068.18	4
3	737.63	8342.61	856.89	8170.53	9080.24	9027.42	9
4	737.63	8349.56	765.14	8315.98	9087.19	9081.12	11
5	737.63	8408.48	737.97	8386.59	9146.11	9124.56	1

Table 6.7: Performance of the complete matheuristic over five runs

The table can be read in the following way. "Step 1" is the optimal solution of the first echelon ("1st E") with the reduced mathematical model on the one hand. The first echelon costs are the same in all runs since they are solved optimally without considering second level routes. On the other hand, the second echelon costs ("2nd E") are calculated by the cheapest insertion and the ALNS-2LR, with 1000 iterations in this case. Next, in "Step 2", the feedback loop is performed, and the first found general improvement is reported. This means that the first echelon costs increased ("1st E") and the second echelon costs ("2nd E") decreased sufficiently to outweigh the higher first level costs. Finally, in "Comparison" the total costs of the first and second echelon before the feedback loop and afterwards are reported. Note that the difference is relatively small, since the first improvement is reported. Even though later it will be tested how much improvement is possible this way in a longer run, the general focus here just lies on proving, that the feedback loop finds an improvement in general and finding out how different cost setting

impact the performance of the algorithm. In this first case, in three out of five runs, four iterations or less of the feedback loop are sufficient to find an improvement. This already shows that solving the problems on both levels jointly, in a two echelon problem, brings additional value since the solution of both echelons separately does not lead to a joint global best. The average run time for 10 iterations of the feedback loop over five runs in this setting was 277.24 seconds.

6.2.3 Runs with different cost settings

Influence of the positioning of the major terminal

In this first set of experiments the second level routing costs were set considerably higher than the ones on the first level. This is due to the fact that all locations, including the one of the major terminal and the DCs were generated randomly. For a supplier of horticultural products in an urban context it is, however, more realistic to assume that the major terminal is relatively far away from all other vertices, thus the distance between DCs and terminal is higher than the distances within the city. For this reason, in the second random data set the distances between customers remain the same as in the previous case. The same is true for the assignments. However, the terminal is placed further away, in the left lower corner of the grid, so at position (0,0). For this setting, the same results as in the previous case are presented in table 6.8. Not surprisingly, the first echelon costs increase while the second echelon costs remain roughly the same. In all five runs that were performed, three runs or less of the feedback loop sufficed to find a global improvement.

Run	Step 1		Step 2		Comparison		
	1st E	2nd E	1st E	2nd E	Total before	Total after	iterations
1	994.14	8369.38	1073.93	8285.80	9363.52	9359.73	3
2	994.14	8449.49	1000.59	8301.49	9443.63	9302.08	1
3	994.14	8342.61	1028.76	8170.54	9336.75	9199.30	3
4	994.14	8359.17	1024.74	8284.31	9353.31	9309.05	2
5	994.14	8378.45	996.24	8317.76	9372.59	9314.00	3

Table 6.8: Performance of the complete matheuristic over five runs with depot at (0,0)

This time it is also assessed, how the first echelon costs elements evolve precisely due to the change of delivery patterns. Table 6.9 gives an overview of the single cost elements in the objective functions over the five runs which were performed. It is noticeable that the first level routing costs, consisting of fuel and driver costs, increased in all five runs compared to the initial values. Order and holding costs changed only slightly compared to the original values and waste costs did not seem to be affected by the changing delivery patterns. The reason for this will be elaborated further later.

	Fuel Costs	Driver Costs	Order Costs	Holding Costs	Waste Costs
Initial Values	562.37	149.52	150.37	49.29	82.60
Run 1	625.34	166.26	147.72	52.00	82.60
Run 2	566.29	150.56	150.10	51.04	82.60
Run 3	589.34	156.69	150.49	49.65	82.60
Run 4	584.74	155.46	151.75	50.19	82.60
Run 5	562.60	149.58	150.21	51.26	82.60

Table 6.9: Development of the cost factors over five runs with depot at position (0.0)

Influence of a changing grid size

The same experiments are next performed on the exact same instance, but on a smaller grid. While a grid size of 100 times 100 (in kilometer) was used previously, now it is reduced to 10 times 10, which changes the cost structure by about halving the first echelon costs, while reducing the second level costs approximately to a tenth of the previous values. As in the previous experiments, the major terminal again lies at position (0,0). In table 6.10 the algorithm performance is reported as for the previous cases:

Run	Step 1		Step 2		Comparison		
	1st E	2nd E	1st E	2nd E	Total before	Total after	iterations
1	350.27	835.05	359.00	819.26	1185.32	1178.26	2
2	350.27	843.99	352.54	823.28	1194.26	1175.82	3
3	350.27	841.27	354.56	832.39	1191.54	1186.95	11
4	350.27	846.67	350.90	842.90	1196.94	1193.80	3
5	350.27	835.99	351.54	832.39	1186.26	1183.93	8

Table 6.10: Performance of the complete matheuristic on instance with grid size 10 x 10

Finding the first improvement seems to take insignificantly longer, which could be explained by the relatively higher first level costs, that make it more difficult to find a general improvement by decreasing second level costs. This is also reflected in the development of the first echelon cost. It seems like an improvement in many cases can be found only if the new delivery patterns do not increase the first level costs too much. While the previous runs found improvements even if the first level costs increased of 3.10 % on average, now it was only an increase of around 1.00 %. On the other hand, the average reduction of the second echelon costs were 1.31 % in the previous case and 1.28 % in the case with the smaller grid, so almost the same. A possible implication is, that the algorithm performs best for a case where the first level costs are relatively small compared to the costs of the second level. In section 6.2.4 this will be studied in more detail.

Table 6.11 again reports the detailed evaluation of the objective function elements.

	Fuel Costs	Driver Costs	Order Costs	Holding Costs	Waste Costs
Initial Values	64.61	17.18	149.56	36.31	82.60
Run 1	65.85	17.51	149.28	37.29	82.60
Run 2	68.48	18.21	150.08	39.63	82.60
Run 3	65.46	17.40	147.30	41.80	82.60
Run 4	64.30	17.09	149.56	36.61	82.60
Run 5	65.46	17.40	149.34	36.74	82.60

Table 6.11: Development of the cost factors over five runs with grid size 10 x 10

Influence of changing waste levels

As the tables in the previous two cases show, there seems to be no change in waste levels for the given instances in different runs. In general, waste levels do not seem to change over different instances so far. The reason is found in the fact, that in all previously tested instances the supply was considerably lower than the demand, which leads to relatively high order quantities. Since the problem is of deterministic nature, the order quantity is determined in a way that no waste at all occurs in all periods except for the first one (in which it is inevitable that the oldest batch of the starting inventory is wasted). This might imply, that the waste level is fixed in general for a deterministic problem and including it in the cost objective is irrelevant for the final model solution.

In order to prove that including waste penalties in the objective function is not obsolete, the reduced mathematical model is solved for a case where supply quantities are increased in a way that they are generally above the demand level. Like this, order quantities reduce to zero and waste costs as well as holding costs increase considerably. As already mentioned, our business partner in interviews stated that in general the company prefers overproduction instead of not satisfying the demand of some customers. Also, for customers ordering from local suppliers, outsourcing might not be welcomed. For this reason it is generally more realistic to assume, that the supply of horticultural products in such a setting will usually exceed the demand.

An additional change undertaken for the next set of experiments is that the maximal lifetime m is reduced from 4 to 3 days, which also increases the waste quantities in the different periods. Please note that for the currently analyzed case with 400 customers with demands between 1 and 100 (where half of them demand this quantity twice a week), the average daily demand is of around 6,000 units.

In the following section, three cases with different supply levels shall reveal the impact of the waste penalties in the objective function to the solution. Table 6.12 gives an overview of the internal supply quantity for all days together with the waste level if a

waste penalty is not imposed compared to a case including waste penalties. Table 6.13 compares the resulting cost factors with and without waste penalty.

CASE 1	**Day 1**	**Day 2**	**Day 3**	**Day 4**	**Day 5**
Supply	9215	9191	4339	5697	5324
Waste without waste penalty	900	679	99	715	2
Waste with waste penalty	900	0	0	0	0
CASE 2	**Day 1**	**Day 2**	**Day 3**	**Day 4**	**Day 5**
Supply	6095	6822	6906	6071	6072
Waste without waste penalty	900	229	0	0	0
Waste with waste penalty	900	0	0	0	0
CASE 3	**Day 1**	**Day 2**	**Day 3**	**Day 4**	**Day 5**
Supply	11215	11191	6339	7697	7324
Waste without waste penalty	900	684	587	3862	530
Waste with waste penalty	900	267	52	795	0

Table 6.12: Supply and waste quantities with and without consideration of waste penalties in the objective function for cases 1, 2 and 3

CASE 1	**Fuel**	**Driver**	**Order**	**Holding**	**Waste**
Costs without waste penalty	62.35	16.58	0.0	73.32	239.5
Costs with waste penalty	62.32	16.57	0.0	81.36	90.00
CASE 2	**Fuel**	**Driver**	**Order**	**Holding**	**Waste**
Costs without waste penalty	62.35	16.58	0.0	73.32	239.5
Costs with waste penalty	61.81	16.43	0.0	79.82	112.9
CASE 3	**Fuel**	**Driver**	**Order**	**Holding**	**Waste**
Costs without waste penalty	67.81	18.03	0.0	279.29	656.30
Costs with waste penalty	70.14	18.65	0.0	340.5	201.40

Table 6.13: Solution of the reduced mathematical model with and without consideration of waste penalties in the objective function for cases 1, 2 and 3

In case 1, the reduced mathematical model was solved for a case where the supply was generated randomly between 4,000 and 10,000 units per day. As table 6.12 shows, the wasted quantity is relatively high if waste is not penalized. Including waste penalties in the objective function, however, changes the waste levels over the planning horizon in a way that only the inevitable waste on day one remains. Please note that there is almost no effect on the first level routing costs, as shown in table 6.13. However, the holding costs seem to increase if a waste penalty is imposed, but this increase is clearly out-weighted by the reduction in waste costs. Order costs remain 0.

The only difference between case 1 and case 2 is, that the supply in the latter is

generated randomly between 6,000 and 7,000, so fluctuations in the harvested quantities are reduced. The results are similar to the first case. As shown in table 6.12, also here including waste penalties leads to the maximal reduction of waste possible, while the cost factors in table 6.13 also evolve similarly to the first case.

Finally, in case 3 the supply is even higher than in the previous cases, namely generated randomly in a range of 6,000 to 12,000. In this case the model is not able to reduce waste to the minimum for all days of the planning period anymore. As shown in table 6.13, in this case the fuel and driver costs again remain relatively stable, the order costs stay 0. As expected, the waste penalty leads to higher holding costs but a substantial reduction in waste. In this case it is clearly visible that the reduction of waste leads to an increase in holding costs and in fact changes the solution. Also the fuel and driver costs increased slightly in the tested setting this time.

The general conclusion of this analysis is that, the higher the supply the more relevant it becomes to include waste in the objective function even in a deterministic case. For low supply and high order quantities it might be irrelevant to include waste quantities in the objective function, since the deterministic model will determine the order quantity in an inventory reducing, and thus waste minimizing, way automatically.

Since the general algorithm so far was only tested for cases, in which the waste levels were minimized automatically, now the previously used instance is adapted to the case with random supply between 6,000 and 12,000 units as in case 3. The algorithm performance is again presented. The increased supply quantity and the reduced maximum lifetime of 3 instead of 4 are the main changes compared the instance in the previous tests.

As indicated by table 6.14, the number of iterations necessary to find an improvement in this case seems to be considerably higher than in the instances used so far.

Run	Step 1		Step 2		Comparison		
	1st E	2nd E	1st E	2nd E	Total before	Total after	iterations
1	632.82	844.57	633.59	841.59	1477.39	1475.18	15
2	632.82	842.35	636.24	838.54	1475.17	1474.78	22
3	632.82	837.88	633.46	835.78	1470.70	1469.24	5

Table 6.14: Performance of the complete matheuristic over three runs with the supply from case 3

Table 6.15 shows how the cost elements changed in the performed three runs. As visible in this tables, the waste level (and therefore the waste costs) still do not change over different runs. This is because of the comparatively high waste penalty of 0.1, which causes that waste is reduced as much as possible because it is relatively more expansive than other operations. However, it changes immediately, if the waste penalty is decreased from 0.1 to 0.01 monetary units, as the results of the six runs with this cost change show, depicted in table 6.16.

The first level cost elements changed as depicted in table 6.17 in the performed 6 runs:

	Fuel Costs	Driver Costs	Order Costs	Holding Costs	Waste Costs
Initial Values	71.56	19.03	0.0	340.84	201.40
Run 1	71.71	19.07	0.0	341.41	201.40
Run 2	70.14	18.65	0.0	346.06	201.40
Run 3	71.97	19.14	0.0	340.95	201.40

Table 6.15: Development of the cost factors over three runs with the supply from case 3

Run	Step 1		Step 2		Comparison			
	1st E	2nd E	1st E	2nd E	Total before	Total after	iterations	runtime
1	421.27	848.85	426.07	840.07	1270.12	1266.14	4	168.89
2	421.27	846.27	427.63	836.86	1267.54	1264.49	19	204.74
3	421.27	845.55	422.12	842.30	1266.82	1264.42	1	163.03
4	421.27	846.85	422.99	844.08	1268.12	1267.07	70	444.20
5	421.27	835.28	423.67	832.64	1256.55	1256.31	7	172.55
6	421.27	843.07	423.61	839.07	1264.34	1262.68	12	240.81

Table 6.16: Performance of the complete matheuristic with the supply setting from case 3 and waste penalties of 0.01 instead of 0.1 per unit

	Fuel Costs	Driver Costs	Order Costs	Holding Costs	Waste Costs
Initial Values	67.84	18.03	0.0	287.83	47.57
Run 1	67.81	18.03	0.0	290.96	49.27
Run 2	70.33	18.70	0.0	288.43	50.18
Run 3	67.81	18.03	0.0	288.70	47.57
Run 4	67.84	18.04	0.0	289.55	47.57
Run 5	69.42	18.46	0.0	288.23	47.57
Run 6	68.84	18.30	0.0	288.58	47.89

Table 6.17: Development of the cost factors over six runs with the supply setting from case 3 and waste penalties of 0.01 instead of 0.1 per unit

6.2.4 Runtime and Performance Analysis

In the final part of the computational study, the performance and the limits of the algorithm shall be tested.

First of all, since all previous tests were performed in a first improvement manner, the algorithm is now run for a given number of iterations. The runtime and the best found solution are reported. These runs are all performed on the last presented instance of the

previous chapter, since this is the most realistic case based on the information we got from our business partner.

It is important to mention here, that this setting was one of those where a relatively high number of iterations were necessary to find a first general improvement. Therefore, it can be assumed that different settings would lead to faster improvements of the solution. Table 6.18 summarizes the results of five runs with 100 iterations of the feedback loop as a stopping criterion:

Run	Costs without 2nd level routing					Results		
	fuel	driver	order	holding	waste	Total	time (s)	improvements
1	70.54	18.75	0.00	299.29	53.42	1251.45	878.58	11
2	67.81	18.03	0.00	288.64	48.25	1260.37	839.56	2
3	68.21	18.14	0.00	297.72	47.39	1259.80	837.42	4
4	67.84	18.04	0.00	300.56	49.17	1254.26	800.27	7
5	67.84	18.04	0.00	293.48	48.68	1262.54	813.50	7

Table 6.18: Performance of the matheuristic (100 iterations of the feedback loop)

In the table, the elements of the objective function for the first echelon are presented for the best found solution together with the total costs of this solution. The runtime in seconds and the number of improvements found in 100 iterations are also reported. Not surprisingly, runs with a higher number of found improvements led to lower costs. While the first improvement criterion for this instance (see table 6.16) led to an average solution cost of 1263.52, in 100 iterations this average was 1257.68.

Table 6.19 gives the same information for five runs with the stopping criterion of 1000 iterations of the feedback loop. In all cases, the total costs could be decreased further compared to 100 iterations.

Run	Costs without 2nd level routing					Results		
	fuel	driver	order	holding	waste	Total	time (s)	improvements
1	69.13	18.37	0.00	291.45	50.17	1242.23	3429.77	15
2	70.54	18.75	0.00	295.34	53.36	1248.84	5361.84	18
3	68.19	18.13	0.00	306.25	46.48	1240.93	3489.15	18
4	70.13	18.64	0.00	300.57	59.32	1243.90	4056.87	20
5	68.21	18.13	0.00	312.30	49.53	1244.37	5583.48	22

Table 6.19: Performance of the matheuristic (1000 iterations of the feedback loop)

6.2.5 Limits of the Algorithm

Finally, the limits of the algorithm shall be tested on the same instance. Since the algorithm improves the overall solution, which means that the first level costs increase and the second level costs decrease, it could eventually happen, that in a case with very high costs on the first echelon and comparatively low ones on the second echelon no improvements can be found by the feedback loop anymore at some point in a reasonable time frame. Whether such limits of the algorithm exist and where they lie is tested by the following part, where different cost factors are changed to see if and how this affects the performance of the presented matheuristic. The results are presented in table 6.20.

A run in which no improvement is found in five runs with 100 iterations of the feedback loop is here defined as a "limit" of the algorithm. If an improvement is found, the tests for the tried cost setting are stopped and it is classified as improvable by the feedback loop and, thus, gets a "no" in the column "limit?".

If an improvement was found it is reported how many iterations of the feedback loop were necessary to find it in the last column "it.". If, however, a limit of the algorithm is reached, no improvement can be reported. In this case only the cost elements of the objective function are displayed in the table for the best found solution in order to give an impression of the cost distribution.

The first set of experiments here is performed by increasing the first echelon routing costs by gradually raising the driver wages on this level. So far, with a wage of 0.001 euro per meter for first level drivers, the first echelon costs ranged between 400 and 450 and the second level costs between 800 and 850, so almost twice as much. Table 6.20 shows how this changes if driver wages per meter are increase to 0.01, 0.1 and even 1. Surprisingly, even raising first echelon driver wages to an unrealistically high level of 1 euro per meter does not make the algorithm less efficient and the number of iterations necessary to find a first improvement does not change considerably.

Secondly, the driver costs were set back to the original (0.001 monetary units per meter) but the inventory holding costs were increased by multiplying the original values with 1.2, 1.5, 2 and 3. In the last mentioned case with extremely high holding costs compared to all other cost factors, the feedback loop does not find improvements anymore. But since it was necessary to increase the holding costs until they were around ten times higher then the sum of all other costs to find this limit, it can be said that the algorithm works well in this case, too.

In the third setting everything was set back to the original but the grid size was decreased step-wise from 5 times 5 to 2 times 2 (in km). The final set, with very limited distances (corresponding to a very limited delivery area of 2 times 2 kilometers), holding costs again become far higher than all other cost factors, which brings the algorithm to its limits. However, since this is a very unrealistic ratio of holding to transportation costs, it seems like the algorithm works well for realistic assumptions.

Finally, the number of DCs and the capacity of cargo bikes is varied to test the algorithm another time. Please note that for the last two instances with 400 customers and only 3 depots, the capacity of the DCs was raised to 2500 in order to make the problem feasible. No limit of the altorithm was reached for any of these cases.

Setting	Step 1		Step 2		1st E					Performance		
1st E wage	1st E	2nd E	1st E	2nd E	fuel	driver	order	holding	waste	limit?	time (s)	it.
0.01	563	848	569	841	54	143	0	330	42	no	293	20
0.1	1831	848	1833	845	53	1406	0	330	44	no	318	18
1	14286	853	14287	852	52	13820	0	369	47	no	463	26
h_costs	1st E	2nd E	1st E	2nd E	fuel	driver	order	holding	waste	limit?	time (s)	it.
times 1.2	479	846	480	844	68	18	0	345	49	no	441	38
times 1.5	566	842	568	835	68	18	0	432	50	no	311	19
times 2	710	840	711	839	70	18	0	571	52	no	293	6
times 3					77	21	0	910	69	yes		
grid size	1st E	2nd E	1st E	2nd E	fuel	driver	order	holding	waste	limit?	time (s)	it.
(5 x 5)	380	425	382	422	34	9	0	291	48	no	385	32
(4 x 4)	371	345	373	342	27	7	0	290	48	no	361	39
(3 x 3)	364	256	364	254	21	6	0	290	48	no	187	12
(2 x 2)					15	4	0	288	47	yes		
(DC#,c_bike)	1st E	2nd E	1st E	2nd E	fuel	driver	order	holding	waste	limit?	time (s)	it.
(5,900)	458	839	458	837	55	14	0	333	56	no	319	20
(5,750)	458	860	458	859	55	15	0	333	46	no	293	6
(4,900)	424	766	425	765	57	15	0	318	35	no	213	11
(4,750)	425	798	425	796	57	15	0	319	35	no	300	45
(3,1200)	629	765	630	754	48	13	0	519	50	no	318	18
(3,1000)	629	816	634	805	50	13	0	513	58	no	348	37

Table 6.20: First improvement found by the matheuristic for different cost settings to test its limits

These experiments generally show, that the feedback loop finds improvements in a reasonable amount of iterations for most cost settings. The only limits of the algorithm that were found are settings with extremely high holding costs compared to the routing costs. However, having holding costs that are ten times and more higher than the routing costs is not very realistic for a company as our business partner. Therefore, it can be stated that the algorithm is applicable to a large range of different problems.

7 Conclusion

The COVID-19 crisis caused a great diversification of the way people approach their groceries. Ordering local organic food, for example from vegetable box suppliers or online farmers' markets, has become more popular than ever before. (Shveda, 2020) This proves also in the fact that more and more supermarkets offer online groceries and new players appear on the Austrian market, as Alfies, Gurkerl and Justmarkt, as well as international competitors as Amazon Fresh and Ocado.

However, the general expansion of e-commerce and home deliveries in urban areas in the past years comes with different costs as increasing congestion, GHG emission, noise, and health risks (Boysen et al., 2021). For companies as our business partner, also food waste is an important issue, since demand as well as the harvested quantity are subject to uncertainty. All these factors can hinder the rising business model of online ordering of locally grown, perishable food items from fulfilling the customers expectations, if not approached appropriately.

In this master thesis a mathematical model was presented which includes as many of the relevant aspects in this context as possible. Therefore, a 2E IRP was presented, which minimizes the (environmental) costs of transportation, inventory holding and waste of a perishable item. The model was designed based on the real-life case of our business partner. One main focus was the modeling of perishability by tracking the age of each harvested item precisely and penalizing the delivery of CTD products as well as food wastage. The company can influence the waste levels by deciding about delivery patterns to the final customers as well as outsourced quantities. Environmental as well as monetary costs of transportation are reduced by organizing the supply chain on two levels. Only the first echelon routes, serving several DCs spread over the city area from the central depot, contributes to GHG emissions by resorting to combustion-engine vehicles. This seems necessary due to the large quantities that are usually transported on these routes as well as the large distances. However, the second level tours from DCs to final customers do not produce any GHGes since they are performed by cargo bikes.

Because of the considerable complexity of the problem, an optimal solution is possible only for small instances in a reasonable time. For this reason, a novel matheuristic based on the one of Rohmer et al. (2019) was proposed. In the first step, a reduced mathematical model including most decisions except for the routes on the second level is solved to optimality. Afterwards, the second level routes are constructed based on the found solution. In a final step, a feedback loop is used to improve the overall solution by changing delivery patterns of final customers.

The solution method was tested on instances with different positioning of the central depot, varying grid sizes (corresponding to differently sized delivery areas) and various waste-related considerations as a changing lifetime of the product as well as different

internal supply levels. Also different numbers of DCs did not bring the algorithm to its limits. It was proven that the feedback loop is able to find improvements of the overall solution for all realistic cases and helps in decreasing overall costs more, the more iterations of it are performed. However, the time taken for this part of the solution method is relatively high compared to the previous steps. Decreasing this by refining some details of the algorithm could be subject of further research. Another thing that should be studied further is the influence of offering several products with different lifetimes on the overall solution. Also an extension of the problem to a case with stochastic demand and supply could be an interesting topic for future research.

In summary it can be said, that this master thesis presented a mathematical formulation for a rising business model that can be applied to urban food supply chains in real-life. Its complex objective function helps reducing negative externalises of home-deliveries and proved successful in decreasing post-harvest loss of horticultural products. It also shows that the explicit inclusion of waste costs into the objective function can make a considerable difference for the environmental impact of a food supply system. An efficient solution method to the problem was proposed and tested on large instances that are not optimally solvable anymore.

8 References

Alinaghian, M., Tirkolaee, E. B., Dezaki, Z. K., Hejazi, S. R., Ding, W. (2021). An augmented Tabu search algorithm for the green inventory-routing problem with time windows. *Swarm and Evolutionary Computation*, *60*, 100802.

Al Shamsi, A., Al Raisi, A., & Aftab, M. (2014). Pollution-inventory routing problem with perishable goods. In *Logistics operations, supply chain management and sustainability* (pp. 585-596). Springer, Cham.

Amiri, S. A. H. S., Zahedi, A., Kazemi, M., Soroor, J., & Hajiaghaei-Keshteli, M. (2020). Determination of the optimal sales level of perishable goods in a two-echelon supply chain network. *Computers & Industrial Engineering*, *139*, 106156.

Amorim, P., & Almada-Lobo, B. (2014). The impact of food perishability issues in the vehicle routing problem. *Computers & Industrial Engineering*, *67*, 223-233.

Anderluh, A., Nolz, P. C., Hemmelmayr, V. C., & Crainic, T. G. (2021). Multi-objective optimization of a two-echelon vehicle routing problem with vehicle synchronization and 'grey zone'customers arising in urban logistics. *European Journal of Operational Research*, *289*(3), 940-958.

Anderluh, A., Hemmelmayr, V. C., & Nolz, P. C. (2017). Synchronizing vans and cargo bikes in a city distribution network. *Central European Journal of Operations Research*, *25*(2), 345-376.

Azadeh, A., Elahi, S., Farahani, M. H., & Nasirian, B. (2017). A genetic algorithm-Taguchi based approach to inventory routing problem of a single perishable product with transshipment. *Computers & Industrial Engineering*, *104*, 124-133.

Bertazzi, L., Coelho, L. C., De Maio, A., Laganà, D. (2019). A matheuristic algorithm for the multi-depot inventory routing problem. *Transportation Research Part E: Logistics and Transportation Review*, *122*, 524-544.

Bertazzi, L., & Speranza, M. G. (2012). Inventory routing problems: an introduction. *EURO Journal on Transportation and Logistics*, *1*(4), 307-326.

Boysen, N., Fedtke, S., & Schwerdfeger, S. (2021). Last-mile delivery concepts: a survey from an operational research perspective. *Or Spectrum*, *43*(1), 1-58.

Breunig, U., Baldacci, R., Hartl, R. F., & Vidal, T. (2019). The electric two-echelon vehicle routing problem. *Computers & Operations Research*, *103*, 198-210.

Coelho, L. C., Cordeau, J. F., & Laporte, G. (2014). Thirty years of inventory routing. *Transportation Science*, *48*(1), 1-19.

Coelho, L. C., & Laporte, G. (2013). The exact solution of several classes of inventory-routing problems. *Computers & Operations Research*, *40*(2), 558-565.

Coelho, L. C., & Laporte, G. (2014). Optimal joint replenishment, delivery and inventory management policies for perishable products. Computers & Operations Research, 47, 42-52.

Coelho, L. C., De Maio, A., Laganà, D. (2020). A variable MIP neighborhood descent for the multi-attribute inventory routing problem. *Transportation Research Part E: Logistics and Transportation Review*, *144*, 102137.

Crama, Y., Rezaei, M., Savelsbergh, M., & Woensel, T. V. (2018). Stochastic inventory routing for perishable products. *Transportation Science*, *52*(3), 526-546.

Cuda, R., Guastaroba, G., & Speranza, M. G. (2015). A survey on two-echelon routing problems. *Computers & Operations Research*, *55*, 185-199.

Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. *Management science*, *6*(1), 80-91.

Ehmke, J. F., Campbell, A. M., & Thomas, B. W. (2018). Optimizing for total costs in vehicle routing in urban areas. *Transportation Research Part E: Logistics and Transportation Review*, *116*, 242-265.

European Commission (2010, October). *Preparatory study on food waste across EU 27.* Publications Office of the European Union.

https://ec.europa.eu/environment/archives/eussd/pdf/bio_foodwaste_report.pdf

Farias, K., Hadj-Hamou, K., & Yugma, C. (2020). Model and exact solution for a two-echelon inventory routing problem. *International Journal of Production Research*, 1-24.

Federgruen, A., Prastacos, G., & Zipkin, P. H. (1986). An allocation and distribution model for perishable products. *Operations research*, *34*(1), 75-82.

Franceschetti, A., Honhon, D., Van Woensel, T., Bektaş, T., Laporte, G. (2013). The time-dependent pollution-routing problem. *Transportation Research Part B: Methodological*, *56*, 265-293.

Gevaers, R., Van de Voorde, E., & Vanelslander, T. (2009). Characteristics of innovations in last-mile logistics-using best practices, case studies and making the link with green and sustainable logistics. *Association for European Transport and contributors*, *1*, 21.

Guimarães, T. A., Coelho, L. C., Schenekemberg, C. M., & Scarpin, C. T. (2019). The two-echelon multi-depot inventory-routing problem. *Computers & Operations Research*, *101*, 220-233.

Hemmelmayr, V. C., Cordeau, J. F., Crainic, T. G. (2012). An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. *Computers operations research*, *39*(12), 3215-3228.

Hiassat, A., & Diabat, A. (2011). A location-inventory-routing-problem with perishable products. In *41st International Conference on Computers and Industrial Engineering 2011* (pp. 130-135).

Hiassat, A., Diabat, A., & Rahwan, I. (2017). A genetic algorithm approach for location-inventory-routing problem with perishable products. *Journal of manufacturing systems*, *42*, 93-103.

Hsu, C. I., Hung, S. F., & Li, H. C. (2007). Vehicle routing problem with time-windows for perishable food delivery. *Journal of food engineering*, *80*(2), 465-475.

Jia, T., Li, X., Wang, N., & Li, R. (2014). Integrated inventory routing problem with quality time windows and loading cost for deteriorating items under discrete time. *Math-*

ematical Problems in Engineering, *2014*.

Le, T., Diabat, A., Richard, J. P., & Yih, Y. (2013). A column generation-based heuristic algorithm for an inventory routing problem with perishable goods. *Optimization Letters*, *7*(7), 1481-1502.

Nahmias, S. (2011). *Perishable inventory systems* (Vol. 160). Springer Science & Business Media.

OECD Fruit and Vegetables Scheme (2020). *Online Sales of fruit and vegetables in Europe*. https://freshfel.org/wp-content/uploads/2021/02/joint-freshfel-europe-oecd-study-on-internet-sales-in-europe.pdf

Onggo, B. S., Panadero, J., Corlu, C. G., Juan, A. A. (2019). Agri-food supply chains with stochastic demands: A multi-period inventory routing problem with perishable products. *Simulation Modelling Practice and Theory*, *97*, 101970.

Osvald, A., & Stirn, L. Z. (2008). A vehicle routing algorithm for the distribution of fresh vegetables and similar perishable food. *Journal of food engineering*, *85*(2), 285-295.

Perboli, G., Tadei, R., & Vigo, D. (2011). The two-echelon capacitated vehicle routing problem: Models and math-based heuristics. *Transportation Science*, *45*(3), 364-380.

Rabbani, M., Farshbaf-Geranmayeh, A., & Haghjoo, N. (2016). Vehicle routing problem with considering multi-middle depots for perishable food delivery. *Uncertain Supply Chain Management*, *4*(3), 171-182.

Rabbani, M., Ramezankhani, M. J., Farrokhi-Asl, H., & Farshbaf-Geranmayeh, A. (2015). Vehicle routing with time windows and customer selection for perishable goods. *International Journal of Supply and Operations Management*, *2*(2), 700-719.

Rahimi, M., Baboli, A., & Rekik, Y. (2017). Multi-objective inventory routing problem: A stochastic model to consider profit, service level and green criteria. *Transportation Research Part E: Logistics and Transportation Review*, *101*, 59-83.

Rohmer, S. U. K., Claassen, G. D. H., & Laporte, G. (2019). A two-echelon inventory routing problem for perishable products. *Computers & Operations Research*, *107*, 156-172.

Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, *40*(4), 455-472.

Shaabani, H. (2022). A literature review of the perishable inventory routing problem. *The Asian Journal of Shipping and Logistics*.

Schenekemberg, C. M., Scarpin, C. T., Pécora Jr, J. E., Guimarães, T. A., & Coelho, L. C. (2020). The two-echelon inventory-routing problem with fleet management. *Computers & Operations Research*, *121*, 104944.

Shveda, K. (2020). How coronavirus is changing grocery shopping. *BBC*. https://www.bbc.com/future/bespoke/follow-the-food/how-covid-19-is-changing-food-shopping.html. (first access date: 27.07.2022)

Song, B. D., & Ko, Y. D. (2016). A vehicle routing problem of both refrigerated-and general-type vehicles for perishable food products delivery. *Journal of food engineering*, *169*, 61-71.

Soysal, M., Bloemhof-Ruwaard, J. M., Haijema, R., & van der Vorst, J. G. (2015). Modeling an inventory routing problem for perishable products with environmental con-

siderations and demand uncertainty. *International Journal of Production Economics*, *164*, 118-133.

Tarantilis, C. D., & Kiranoudis, C. T. (2001). A meta-heuristic algorithm for the efficient distribution of perishable foods. *Journal of food Engineering*, *50*(1), 1-9.

Toth, P., & Vigo, D. (Eds.). (2014). *Vehicle routing: problems, methods, and applications*. Society for Industrial and Applied Mathematics.

Uchoa, E., Pecin, D., Pessoa, A., Poggi, M., Vidal, T., Subramanian, A. (2017). *New benchmark instances for the capacitated vehicle routing problem*. European Journal of Operational Research, *257*(3), 845-858.

Violi, A., Laganá, D., Paradiso, R. (2020). The inventory routing problem under uncertainty with perishable products: an application in the agri-food supply chain. *Soft Computing*, *24*(18), 13725-13740.

Wang, X., Wang, M., Ruan, J., & Zhan, H. (2016). The multi-objective optimization for perishable food distribution route considering temporal-spatial distance. *Procedia Computer Science*, *96*, 1211-1220.

Xavier I., Uchoa, E., Pecin D., Pessoa A., Poggi M., Vidal T., Subramanian A. (2022). Capacitated Vehicle Routing Problem Library.
http://vrp.galgos.inf.puc-rio.br/index.php/en/ (first access date: 24.03.2022)

Yahia, E. M., Fonseca, J. M., & Kitinoja, L. (2019). Postharvest losses and waste. *Postharvest technology of perishable horticultural commodities*, 43-69.

Zhao, Q. H., Chen, S., & Zang, C. X. (2008). Model and algorithm for inventory/routing decision in a three-echelon logistics system. *European Journal of Operational Research*, *191*(3), 623-635.