Heuristics for Tour Planning of Last-Mile Parcel Deliveries by Cargo Bikes

Institute for Transport Studies
Department of Civil Engineering, Geo and Environmental Sciences
Karlsruhe Institute of Technology

Bachelor Thesis

of

Alexander von Heyden

Matriculation Number 1941715

Reviewer:
Prof. Dr.-Ing. Peter Vortisch
Second reviewer:
Prof. Dr. rer. nat. Peter Sanders

Advisors:
M.Sc. Jelle Kübler

Date of submission: 25.04.2022
Karlsruhe

Declaration

I declare that I have developed and written the enclosed thesis without the assistance of others, and have not used sources or means without declaration in the text. External assistance includes paid services of third parties, e.g. Internet agencies, which check the written version for spelling and / or punctuation, grammar, formulation, logic, plausibility, sentence structure, sense or style. This work has not yet been submitted to another examination institution neither in the same nor in a similar way and has not yet been published. This thesis was carried out in accordance with the Rules for Safeguarding Good Scientific Practice at Karlsruhe Institute of Technology (KIT). Furthermore I assert that the electronic version is identical to the print version. I hereby grant the Institute for Transport Studies the right to use this thesis. This includes in particular the further use of the acquired knowledge and results in other works or publications by employees of the Institute for Transport Studies. I also grant the reference library of the Institute the right to use this thesis.

Karlsruhe, 25.04.2022

Alexander von Heyden

Problem statement

During the last years the usage of online shops rose significantly. To meet this increased demand the amount of delivery vehicles had to keep pace. The most common way for last mile parcel deliveries are small trucks with a gross vehicle weight of no more than 3.5 t. Since those cenventional vehicles produce comparatively high environmental pollution and, especially in densly populated areas, exacerbate the lack of parking spaces, transport planners and delivery service companies are interested in alternative delivery concepts.

One such concept is the combination of cargo bikes with micro depots for parcel deliveries. Due to their low emissions and space requirement cargo bikes can be a preferred alternative to traditional delivery vehicles. Considering the vastly decreased capacity and speed of cargo bikes compared to trucks, one has to plan their tours with regards to an entirely different set of restraints. The reduced speed might also increase the travel time for short distance trips, and deliveries to areas further away will become more expensive. Here it might be useful to consider collecting parcels for several days in a depot and deliver them all during the same tour.

Assignment of tasks

In this thesis a heuristic for planning tours of parcel deliveries with cargo bikes shall be developed. Using an existing TSP-solver a tour containing all parcels shall be calculated. Then a greedy algorithm will be designed that partitions this route into smaller, for cargo bikes optimised tours. This algorithm has to consider that all parcels must be delivered within a predetermined time period, ranging from a few days to a week.

Subsequently the algorithm shall be integrated into the already existing travel demand and parcel model *logiTopp* to simulate the delivery of parcels and to study the impact on the traffic the developed algorithm has. For the simulation data from the region Karlsruhe is available.

In this thesis an appropriate transformation shall be chosen, that models delivery requests as a TSP instance. Different selection strategies for the greedy algorithm shall be tested and evaluated. As a last step the developed algorithm shall be compared to the implementation of the current algorithm, using suitable parameters, that evaluate the efficiency as well as the impact on the traffic.

Abstract

Last-mile parcel delivery is concerned with planning the delivery from a central parcel depot to the recipient of the parcel. To deliver those parcels different approaches are possible, one of which is the usage of cargo bikes. Since using cargo bikes introduces a new set of restraints compared to trucks, new methods to calculate the tours are necessary. Nevertheless, there exists no clear best strategy to calculate those tours.
We focus on the concept of micro depots, small depots located around the delivery region that a truck delivers at the beginning of the day, at which cargo bikes can load new parcels to continue delivering without having to return to the main depot. We present ideas to optimise the distribution of these micro depots in the hope of increasing the efficiency of last-mile parcel delivery.
On simulations in the region of Rastatt, we can see that using micro depots increases the quality of the tours.

Zusammenfassung

Last-mile parcel delivery beschäftigt sich mit der Planung von Paketauslieferungen von einem zentralen Depot zu den Empfängern dieser Lieferungen. Um diese Pakete auszuliefern sind verschiedene Ansätze möglich, unter anderem die Verwendung von Lastenfahrrädern. Die Benutzung von Lastenfahrrädern führt zu neuen Herausforderung bei der Planung von Routen und neue Methoden zur Planung sind erforderlich. Eine eindeutig beste Strategie zur Planung existiert nicht.
Im Mittelpunkt dieser Arbeit steht das Konzept der Micro Depots. Bei Micro Depots handelt es sich um kleine Depots, die im Liefergebiet zu Beginn des Tages von LKW ausgefahren werden. An diesen können die Lastenräder während ihrer Tour neue Pakete aufnehmen, ohne zum Hauptdepot zurückkehren zu müssen, wodurch sie effizienter ausliefern können. Diese Arbeit stellt Ideen vor, um die Platzierungen dieser Micro Depots zu optimieren, um die Effizienz der last-mile Lieferungen zu erhöhen.
Unter Benutzung der neuen Methoden können wir mit Simulationen in der Region Rastatt eine Erhöhung der Tourenqualität feststellen.

Contents

List of Figures **ix**

List of Algorithms **xi**

List of Tables **xiii**

List of Abbreviations **xv**

1 Introduction **1**
1.1 Motivation 1
1.2 Contribution 2
1.3 Structure of the thesis 2

2 Preliminaries **3**
2.1 Basics 3
2.2 Modelling framework 5
2.2.1 mobiTopp 6
2.2.2 logiTopp 6
2.2.3 Delivery policies 7

3 Previous Work **11**
3.1 A basic approach 11
3.2 The first tour 15
3.3 Two small bugs 17

4 Approach **19**
4.1 Pick-ups 19
4.2 Micro depots 20
4.3 Truck optimisation 23
4.4 Local TSPs 25

5 Evaluation **29**
5.1 Rastatt 29
5.2 Of trucks and bikes 30
5.3 Micro depots & TSPs 33

6 Conclusion **35**
6.1 Future work 35

Bibliography **37**

List of Figures

2.1 A cargo bike with four wheels in Vienna [Tischbeinahe, 2021] 4
2.2 Possible sequence of steps in a parcel order model, taken from [Kübler und Reiffer, 2022] 8
3.1 Basic *logiTopp* tours without restraints 16
4.1 Relationship between deliveries and pickups 20
5.1 Rastatt with outskirts, from [OpenStreetMap, 2022] 29
5.2 Some of the zones around Rastatt, data taken from [Kübler, 2021] . . . 30

List of Algorithms

1 ZoneTSP.assignParcels 12
2 ZoneTSP.planZoneTour 14

3 ZoneTSP.MicroDepots.assignParcels 22
4 findNextPickup 25
5 localTSP 27

List of Tables

5.1 Comparison between trucks and cargo bikes, single depot, close zones only, average zone distance 31

5.2 Comparison between trucks and cargo bikes, single depot, all zones, average zone distance 32

5.3 Amount of tried deliveries in Rastatt, close zones 32

5.4 Routing with micro depots, close zones, limited delivery agents 33

5.5 Routing with micro depots, close zones, limited delivery agents 34

List of abbreviations

CEPSP	"Courier, Express and Parcel" service providers
TSP	Travelling salesman problem
DAB	DeliveryActivityBuilder
PAB	PickupActivityBuilder
DC	delivery centre
DA	delivery agent

1 Introduction

1.1 Motivation

During the last decade the popularity of online shopping rose to an all time high [Statista, Union]. This trend can primarily be attributed to two factors:
First of which is the ability to choose from a much larger selection of products, ranging from common to very niche products which are usually not in stock in local shops. Paired with the ability to order at any time from most devices connected to the internet this provides a massive boost in convenience for most consumers.
It is also interesting to note that the price of expendable goods tends to decrease compared to the average hourly wage, since companies try to produce as cheap as possible. This in turn also feeds back into an increased amount of purchased products.
The COVID-19 pandemic in 2020 solidified this trend. The curfew led to many local shops closing down, with some starting to provide online services themselves, while others went bankrupt and had to close forever.
This shift in behaviour leads to a new set of challenges, which need to be solved for more efficient parcel delivery. Delivery trucks, currently the most common way of parcel delivery, have a comparatively high impact on the environment. This manifests in an increase in pollution but also in the blocking of parking spaces. They are also responsible for halting traffic when stopping to deliver parcels, which is a major problem, especially in crowded areas. The increased demand can be solved by increasing the size of the delivery truck fleet, but using more trucks also doubles down on all the issues this mode of transportation already has. We are already witnessing a switch to electric powered trucks, which solves some, but not all of the problems described above, can we do better?

The *mobiTopp* framework and its extension *logiTopp* provide an agent bases approach to this problem. Every resident and every delivery person is modelled as an agent, who can interact with each other to simulate a real world environment. This allows an exhaustive evaluation of different approaches and models.

1.2 Contribution

One such concept - and the concept we will examine in this thesis - is the usage of cargo bikes in combination with micro depots. Micro depots can be placed at strategic locations along the route of cargo bikes, so that the bikes can refill parcels without having to drive back to the main depot, thus saving valuable time.
We will look at different combinations of this idea. A fleet of only cargo bikes with varying capacity, as well as cargo bikes being supported by a small fleet of delivery trucks that either distribute micro depots in the morning or act as micro depot themselves.
Furthermore we will also examine the impact of different TSP solvers on the speed and quality of the calculated routing and the granularity of a solution that meets our standards.

1.3 Structure of the thesis

In Chapter 2 we will describe the used frameworks, how they work and which interfaces they provide. We also define key terms used in this thesis. Chapter 3 gives a short overview of the current state of a trivial calculated route, where we use the same restraints for routing cargo bike as with trucks. It also examines the advantages and possible disadvantages of such an approach. The next chapter, Chapter 4, focuses on our approach to the problem presented in the section above. Chapter 5 evaluates the results found in this thesis and Chapter 6 concludes it while pointing out possibilities for future work.

2 Preliminaries

This chapter explains the ideas and concepts used within this thesis. We start by outlining the properties of cargo bikes, review common definitions for problems and terms, and give a detailed overview of mobiTopp and logiTopp, the frameworks which we will be focusing on.

2.1 Basics

Last-mile parcel delivery refers to the final step in the delivery chain, responsible for delivering parcels to the recipient's address. The use of cargo bikes takes on a central point in this step and if the approach used in chapter 4, it is therefore paramount to have a good understanding of the properties and limitations of such a bike.

Definition 1 (Cargo bikes)**.** *Compared to normal bicycles, cargo bikes are built with a sturdier frame, are wider and longer and excel in their ability to carry large amounts of goods. Depending on the requirements needed, a cargo bike can have between two to four wheels, an auxiliary electric motor and cargo space as needed. The cargo is usually either stored in a wide compartment between the front- or back wheels in case of a bike with three wheels, or, in case of a bike with two wheels, in the elongated compartment between the handlebar and the front wheels.*

While a two wheel configuration has increased mobility, allows for easier driving for the untrained rider, and has faster speeds even on narrow bike lanes, we are mostly interested in the variant with three or even four wheels, since the increased stability and cargo capacity are the most relevant statistics for this project.
An example of such a bike is the *City-Hub* (see Figure 2.1), that has recently been tested by DHL, a German logistics company, in major European cities. The *City-Hub* has a swappable cargo container that has a volume of up to 1 m^3 and a maximum load of 125 kg [Reichel].

Before we can deliver parcels, we have to consider the order in which the parcels are to be delivered. To order the deliveries we will model a *Travelling salesman problem* (TSP), a well known problem in theoretical computer science.

Figure 2.1: A cargo bike with four wheels in Vienna [Tischbeinahe, 2021]

The travelling salesman problem is a problem from the field of combinatorial optimisation. It is easy to understand and visualise, but calculating an exact solution is NP-hard [Korte, 2008], which is why we will look at different heuristics and optimisations for solving TSP instances in the following chapters.

Definition 2 (Complete weighted graph)**.** *A complete weighted graph G is an ordered pair (V,E) with finite vertex set V, edge set*

$$E = V^2 := \{(u,v) \mid u,v \in V, u \neq v\}$$

and edge weight function

$$c : E \to \mathbb{R}_+, e \mapsto c(e).$$

Definition 3 (Hamiltonian cycle)**.** *A cycle in a graph that contains each vertex exactly once is called a Hamiltonian cycle.*

Definition 4 (*Travelling salesman problem* (TSP))**.** *For a complete weighted graph $G = (V,E = V^2)$ as in definition 2 the solution to the travelling salesman problem is the Hamiltonian cycle H which minimises the sum*

$$\sum_{e \in E(H)} c(e)$$

of its edge weights, where $E(H)$ is the set of edges in H.

As this is an NP-hard problem, finding exact solutions needs an impractical amount of time for most relevant problems. In later chapters we will therefore consider algorithms that approximate exact solutions to this problem.

Before we start explaining the framework, we introduce some minor definitions which are needed to understand the following sections.

Definition 5 (Delivery)**.** *A delivery is defined by the following attributes:*

- *destination of the delivery*
- *type of the destination (is the delivery agent (DA) delivering the parcel to the recipient's home address, work address or a parcel locker)*
- *delivery centre (DC) from which the last-mile delivery takes place*
- *set of parcels destined for the location*
- *arriving time at the delivery centre, which dictates on which day the first delivery can be tried*

The exact definition of delivery agents and delivery centres will be part of Section 2.2.2.

Since a given parcel can vary in size, weight and form, we use a statistical model for our simulation. Instead of varying parcel sizes, each parcel has a unit size and the size of the vehicle is randomly drawn from a normal distribution. For each type of vehicle the distribution can be parametrised individually.

Definition 6 (Zone)**.** *Given a not necessarily connected area A on a map, a zone Z is a sub area of A.*

Definition 7 (Zonemap)**.** *A partition of a not necessarily connected area A on a map into zones $Z_i, i \in I$ is called a zonemap if and only if the following are true:*

- *For every location $l \in A$ there exists an $i \in I$ such that $l \in Z_i$*
- *For any $i, j \in I, i \neq j$ the set of shared locations $Z_i \cap Z_j$ is the empty set.*

2.2 Modelling framework

To better understand the effect our approach has on last-mile parcel deliveries, we will implement these changes into the modelling framework *mobiTopp* and its last-mile logistics extension *logiTopp* [Reiffer u. a., 2021a,b]. The following sections provide a detaild overview over the used modelling framework.

2.2.1 mobiTopp

mobiTopp [Briem und Kübler, 2021] is a travel demand modelling framework [Mallig u. a., 2013, Mallig und Vortisch, 2017], that consists of a separate long and short term module.
The *long term* module is used to generate a synthetic population according to a given set of restraints. The module can generate households and residents, which are modelled as independent agents. Each agent is assigned a list of socio-demographic attributes, such as age, gender, work status, degree of education, income, place of work/education, drivers licence, commuter ticket, membership of mobility service providers such as car/bike sharing and more. A household groups together a given number of residents and is further assigned the number of cars its members have access to, its location and its gross household income. The household and person data is drawn from the German mobility panel [Ecke u. a., 2020], a national household travel survey, with weighting according to the general population distribution.
For each agent a personal schedule is created that contains all the agent's activities over the course of the simulated period, in this case one week. The schedule might contain activities such as work, education, shopping, leisure, service and home activities. This module is run only once and the data generated can be reused for further simulations.

The *short term* module is where the generated schedules are simulated. For any newly started activity the module picks a location and a mode of transport to this location. If an agent were to go shopping, the module decides to which shop the agent goes and which mode of transport they use. The framework has an estimated duration for all activities and trips, but due to unforeseen changes in the simulation, such as delays in earlier states, the actual time taken in the simulation might differ, which is why the activity schedule of an agent has to be updated after every trip, to account for updated travel times. The agents all act simultaneously and can influence each other. This loop is the repeated for every following activity.

2.2.2 logiTopp

logiTopp [Kübler und Reiffer, 2022] is a logistic extension to the travel demand modelling framework *mobiTopp* [Reiffer u. a., 2021b]. Building on the simultaneous simulation and the possibility of agent-agent interaction found in *mobiTopp*, *logiTopp* models the parcel orders of residents within the population and simulates their last-mile delivery. Since *mobiTopp* uses an agent based model, each resident is an individual agent and can be interacted with.
Most features of *logiTopp* only impact the *short term* module of *mobiTopp* and as such can work on an existing population. Before we run the *short term* module, we pick

agents from the population and assign them the role of delivery agents as well as one of the delivery centres that are part of the model. This guarantees us, that both the agent's private and commercial trips are simulated.
The model is aware of all the socio-demographic attributes of the agents in the population, that have been generated by the *long term* module of *mobiTopp*, and as such can consider all of these at any given time during the process. After assigning the delivery agents, we use the parcel demand model to determine the amount of parcels any given resident in the population expects within the simulated period of one week.
For each parcel the model decides upon a delivery location, drawing from a discrete choice model similar to [Reiffer u. a., 2021a,b]. *logiTopp* supports three types of delivery locations: home, work and parcel locker, one of which is also assigned to the parcel. Furthermore each parcel gets assigned a *"Courier, Express and Parcel" service providers* (CEPSP) - which, among other things, determines the delivery policies - a delivery centre from which the parcel will ship and the date it arrives at the delivery centre. The order in which these are assigned is customisable, and can depend of previous assigned attributes. Figure 2.2a is a possible sequence of these assignments. The model assumes a continuous uniform distribution on parcel arrivals at the delivery centre, except on Sundays, on which no parcels are delivered in Germany.

Once this part of the parcel demand model has finished, the simulation continues in *mobiTopp*. Every time a delivery agent arrives at their assigned delivery centre, *logiTopp* computes a tour and assigned parcels to the respective delivery person, a visualisation of the following process can be seen in figure 2.2b.
The default implementation uses a route first, cluster second [Beasley, 1983] approach, solving a giant TSP instance and splitting it into pairwise disjunct tours for each delivery agent. By default a 2-approximation provided by the graph library JGraphT [Naveh, 2022] is used to solve the TSP, in later chapters we will look at using different solvers. After the tour is assigned to the delivery agent it is split again into a list of simple delivery activities, which contain all pending parcels in the correct order. These delivery activities are encapsulated between a *load* and *unload* activity.
Before control is handed back to the *mobiTopp* framework, the activities are added to the agent's activity schedule, so that they can be performed simultaneously to the other activities planned by *mobiTopp*. Since all agents are simulated simultaneously by *mobiTopp*, the location of all agents is known at the time of the delivery, and it is thus easy to verify, whether a delivery is successful or not.

2.2.3 Delivery policies

As mentioned in section 2.2.2, every CEPSP has its own delivery policy. This policy can have a vast influence on the results of the simulation.

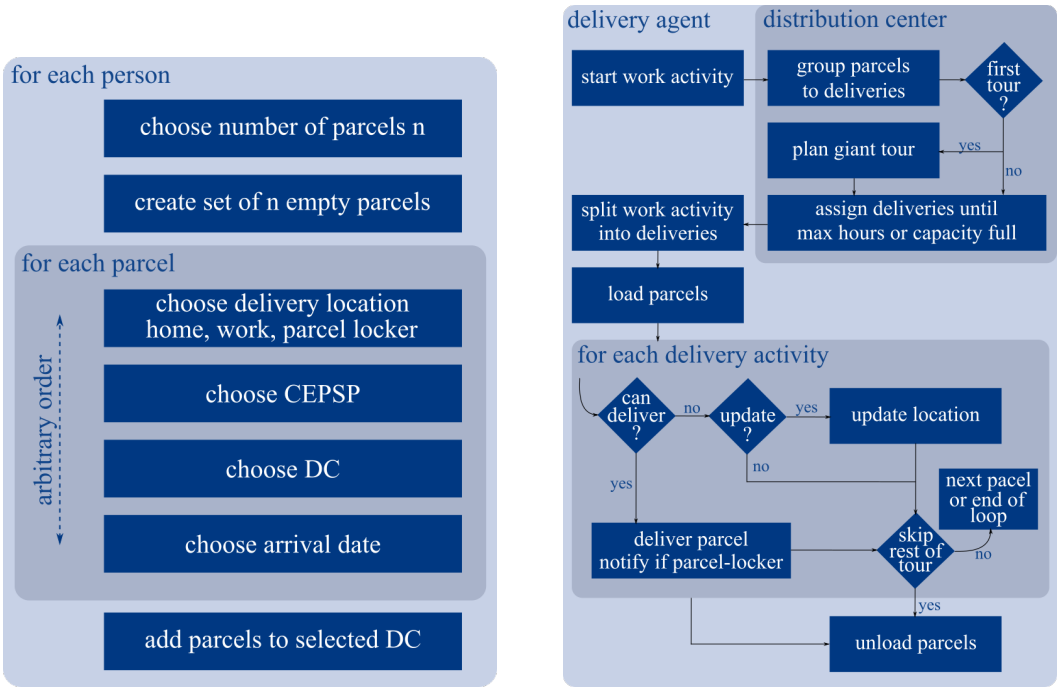


Figure 2.2: Possible sequence of steps in a parcel order model, taken from [Kübler und Reiffer, 2022]

"Courier, Express and Parcel" service providers (CEPSP)
delivery centre (DC)

Any CEPSP has a time window in which it may deliver parcels. This does not only affect the work time of its agents, but attempts at different times during the day might have varying success rates. During the late afternoon residents are more likely to be at home than during the morning, where most are at work. A CEPSP might try up to α_{max} delivery attempts, before deciding to deliver the parcel to a parcel locker, at which a delivery is always successful, The current model does not model parcel lockers, at such their capacity is assumed to be infinite. If α_{max} unsuccessful deliveries have taken place, the parcel is placed in a parcel locker and the recipient is informed who now schedules a pick up activity in front of their next home activity.
A delivery policy also adds restraints for successful deliveries. If it is a *work* delivery, the current model enforces a personal delivery, so the recipient has to be at work. If it is a *home* delivery, the model tries to deliver to household members or neighbours if the recipient is not available, which is possible since *mobiTopp* is agent based. This can be done by accessing the recipient's household context available on *mobiTopp*. As outlined above *parcel locker* deliveries are used as fallback delivery locations, but are also a valid location in the first place.

3 Previous Work

This chapter gives a small overview of the default implementation used to deliver parcels with trucks and acts as a stepping stone to chapter 4, where we will build upon this chapter.

3.1 A basic approach

Let us first examine Algorithm 1 used in [Kübler und Reiffer, 2022] to generate a basic tour that contains all required delivery destinations. The algorithms assumes all sets to be strictly ordered. As seen in line 3 the algorithms requires 4 arguments.
deliveries is a collection of all parcels that are available at the main depot and are ready for shipment to their destination.
The *person* object contains information that mostly influences the speed at which parcels are delivered in two ways. A faster driving person spends less time driving between deliveries, whereas a person's efficiency determines how long it takes this person to deliver a parcel, once it has reached its location.
currentTime is the time the algorithms should assume when planning the route. It directly impacts the travel time between locations, since the same route may take longer during different time periods of the day, such as the evening commute where workers are returning home.
remainingWorkTime is the remaining time the delivery agent, whose route this will be, has left. A shorter time frame leads to a shorter assigned route.
Since this is not the simulation, but just the precalculation, it is not guaranteed that for all assigned parcels a delivery attempt will be made, since delays during the actual delivery might lead to the delivery person finishing work, while still having assigned parcels left over. These parcels will be carried over to the next day and be part of that day's giant TSP tour.

The *if-condition* in line 4 checks whether a new TSP route has to be calculated. This is always the case if *assignParcels* is called for the first time any given day, or if the last TSP solution is older than one hour. This approximates the change in travel time observed between similar locations during the day. Later calls of *assignParcels* might therefore be able to skip the expensive TSP calculation.

Lines 8 to 12 setup the required environment, *selectCapacity(person)* decides upon the available cargo space of *person* by drawing from a normal distribution as mentioned in chapter 2.1.
The assignment of parcels to a tour takes place in lines 13 to 27. From *parcelTour*, which is a TSP tour of all parcel destinations, we continue adding the next delivery on the route to the planned tour, *assignedParcels*, if doing so is still within the transport capacity of the delivery person and we estimate that the delivery person has sufficient remaining work time, that they could return to the depot after this delivery.
If either is not the case we remove all during this run assigned parcels from *parcelTour*, so that they will not be considered in future runs with the same TSP route, and return the ordered collection of parcels as the result of *assignParcels*.

Algorithm 1 ZoneTSP.assignParcels

```
 1: lastPlannedTime ← Time.start
 2: parcelTour ← {}
 3: procedure ASSIGNPARCELS(deliveries, person, currentTime, remainingWorkTime)
 4:     if parcelTour is outdated then
 5:         parcelTour ← planZoneTour(person, deliveries, currentTime)
 6:         lastPlannedTime ← currentTime
 7:     end if
 8:     assignedParcels ← {}
 9:     capacity ← selectCapacity(person)
10:     lastZone ← person.distributionCenter.zone
11:     time ← currentTime
12:     endOfWork ← currentTime + remainingWorkTime
13:     for all delivery ∈ parcelTour do
14:         if assignedParcels.size ≥ capacity then
15:             break
16:         end if
17:         tripDuration ← travelTime(person, delivery.zone, lastZone, time)
18:         deliveryDuration ← delivery.estimateDuration(person.efficiency)
19:         time ← time + tripDuration + deliveryDuration
20:         returnDuration ← travelTime(
                person, delivery.zone, person.distributioncenter.zone, time)
21:         if time + returnDuration ≤ endOfWork then
22:             assignedParcels.add(delivery)
23:         else
24:             break
25:         end if
26:         lastZone ← delivery.zone
27:     end for
28:     parcelTour.removeAll(assignedParcels)
29:     return assignedParcels
30: end procedure
```

Algorithm 2 shows the *planZoneTour* function used in the previous algorithm. The argu-

ments are mostly identical, only *remainingWorkTime* is missing, since this information is not needed to solve a TSP. This algorithms considers all zones which have pending deliveries and calculates a shortest tour connecting all deliveries.
Line 2 to 16 transform the deliveries into a TSP instance which we solve using a black-box solver. Line 3 in particular guarantees, that the zone of the delivery centre is a vertex in the TSP instance. For each delivery its zone is mapped to a vertex, if multiple deliveries are located within the same zone, their zones are mapped to the same vertex. The *distance* call provides a very simple approximation for the distance between two zones. A lookup table provides the average distance between any two zones, as a side effect of this approximation any pair of two locations from two not necessarily different zones share the same distance.
Once we have the zone TSP we need to find the starting point of the tour. One of the vertices in *tspTour.vertices* is the zone in which the delivery centre is located, and thus is the beginning of the tour. The code in lines 20 to 31 does exactly this; It "cuts" the vertex list in two parts, *prefix*, a list that starts with the zone that contains the delivery centre, followed directly by the zones that come after it in the TSP vertex list (*tspTour.vertices*) and *suffix*, a list that starts with the first zone in the original list up to the zone directly in front of the delivery centre. Line 24 in particular contains a lambda expression, that filters all the deliveries that are to be delivered in the current zone.
What remains is taking the now ordered zone list and add all the deliveries back in. Using the approximation mentioned above, the order in which we add the deliveries does not matter, as long as we add all deliveries from any one zone before adding the deliveries from a second zone. Instead of adding zones to our *prefix* and *suffix* list, we can add all the deliveries in the current zone. This keeps the ordering of the zones and guarantees that all deliveries are added. We can therefore see that line 32 returns a valid result.

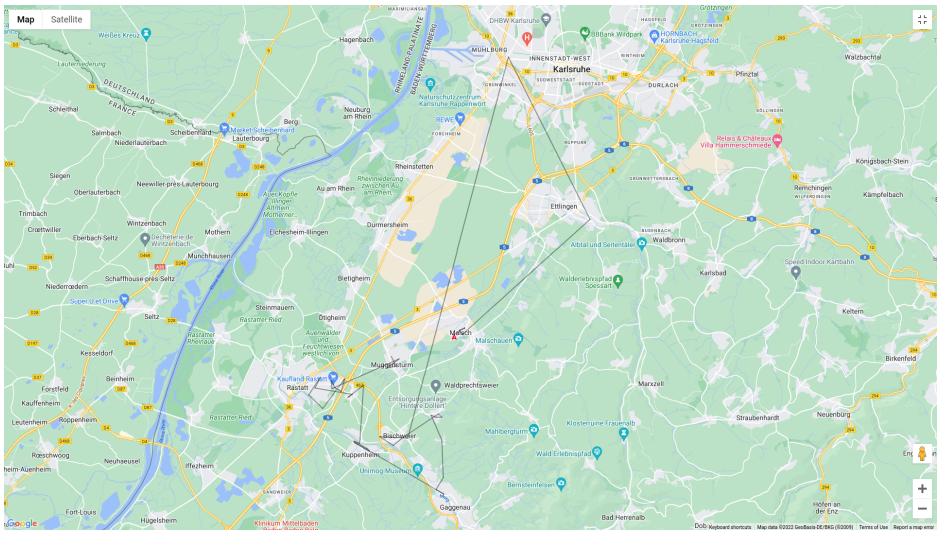
Algorithm 2 ZoneTSP.planZoneTour

```
1: procedure PLANZONETOUR(person, deliveries, currentTime)
2:     graph ← simpleWeightedGraph
3:     start ← person.distributionCenter.zone
4:     graph.addVertex(start)
5:     for all delivery ∈ deliveries do
6:         if delivery.zone ∉ graph.vertices then
7:             graph.addVertex(delivery.zone)
8:         end if
9:     end for
10:    for all zone_1 ∈ graph.vertices do
11:        for all zone_2 ∈ graph.vertices\{zone_1} do
12:            edge ← graph.addEdge(zone_1, zone_2)
13:            edge.setWeight(distance(zone_1, zone_2, currentTimea))
14:        end for
15:    end for
16:    tspTour ← TspSolver.solve(graph)
17:    prefix ← {}
18:    suffix ← {}
19:    foundStart ← false
20:    for all zone ∈ tspTour.vertices do
21:        if zone = start then
22:            foundStart ← true
23:        end if
24:        for all delivery ∈ {delivery ∈ deliveries | delivery.zone = zone} do
25:            if foundStart then
26:                prefix.add(delivery)
27:            else
28:                suffix.add(delivery)
29:            end if
30:        end for
31:    end for
32:    return prefix + suffix
33: end procedure
```

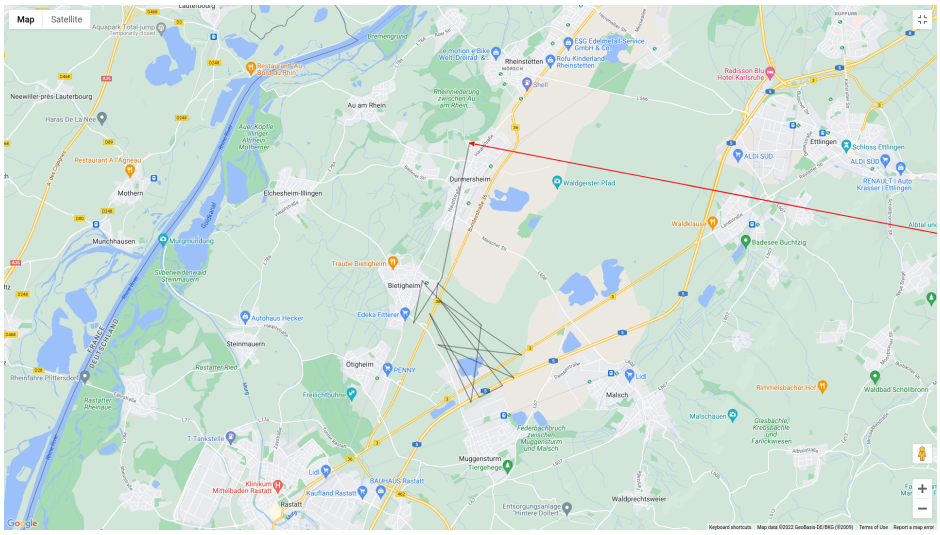
3.2 The first tour

Using our current knowledge, we can use *logiTopp* and *mobiTopp* to plan a tour that currently has no restraints. To visualise the output, we use a tool developed to plot *mobiTopp* activities [Paul019, 2021].
logiTopp provides a scaling option, that scales the amount of ordered parcels down and decreases the amount of delivery agents by the same factor to improve performance. Exact numbers will be given in Chapter 5.2, but reducing the amount of parcels by 90 % can decrease the simulation time by over an order of magnitude.
Figure 3.1 shows three routes, each of which have been assigned to a delivery agent in a different simulation. We can see that the routes are quite similar in length, which leads to the hypothesis that the limiting factor of trucks is not their cargo space, but the limited amount of time the delivery agent is working.
We can further notice, that the routes intersect themselves, especially in Figures 3.1a and 3.1c, which is untypical for solutions to a TSP. This can be explained by the approach we have taken on computing the tour: Since we use the actual locations of the deliveries to plot the tour, but only solve the TSP on the zone layer (see Figure 5.2), and add deliveries within zones in - essential - random order, we end up with tours that cross themselves within zones.

(a) Scaled down to 1 %



(b) Scaled down to 10 %



(c) No scaling

Figure 3.1: Basic *logiTopp* tours without restraints

3.3 Two small bugs

In the implementation of Algorithm 1 two bugs are present that have been fixed in the provided pseudocode. An evaluation of these bugs and the actual impact they have can be found in Chapter 5.2.

Missing time update The original code is missing line 6 in Algorithm 1. This line is responsible for updating the time, at which the last tour calculation took place. Since this time stamp is never updated, the algorithms has to recalculate a TSP every time it is called. This may lead to unnecessary overhead, since solving TSPs is expensive. Depending on the time resolution of travel costs it could lead to better results.

Missing zone update The original code is also missing line 26, which is responsible for updating the zone after an expected successful delivery has been added to the precalculated route. The starting value of *lastZone* is the zone of the delivery center, which is then continuously used for distance calculations. This has no effect on the order in which parcels are delivered, but it could drastically influence the quality of the result.

Two extremes are possible: If the delivery person is currently far away from the delivery center, the estimated distance - and therefore travel time - to the next delivery is far bigger than it should be, which might lead to the algorithm only assigning very few parcels to the delivery person, because the remaining work time reduces too fast. We thus estimate the required amount of delivery persons to be greater than in reality.

On the flip side it could happen that the distances we assume are far to short, and we assign so many parcels to a single agent, that it is impossible for them to deliver them all, even in perfect conditions. This might lead to a decreased amount of delivery persons, but it also increases the amount of parcels that return without a delivery attempt. This negatively affects the successful delivery quotient and adds more parcels to the following day.

4 Approach

This chapter explains the approach we take to implement micro depots into *logiTopp*. Micro depots are small depots that can be placed at strategic positions along the route of cargo bikes, so that the bikes can resupply without having to drive back to the main depot. Continuing we will also take a look at basic optimisation of the location of these micro depots, which could possibly be used to merge micro depot locations or reduce the distance the trucks have to drive to ship out the micro depots to their location for the day.
We will also implement access to different TSP solvers and solve local TSPs within zones which should reduce the amount of zone internal crossings that can be found in Figure 3.1, and should also lead to better results if we switch from average distance between locations to an approximate distance based on their latitude and longitude.

logiTopp provides a configuration file that allows to set and change provided parameters before runtime. We will extend this configuration file, so that every addition we make to the code can either be configured or completely disabled. Disabling all options will result in Figure 3.1.

4.1 Pick-ups

Before we can start to implement the logic behind micro depots, we have to introduce the concept of pick-ups, which model the process of a delivery person moving parcels from a micro depot onto their cargo bike. During the simulation, every time a delivery agent arrives at a destination, a *DeliveryActivity* is executed, that determines, among other things, the exact amount of time spent at this location. As seen in figure 4.1, this *DeliveryActivity* is generated by a *DeliveryActivityBuilder (DAB)*. Besides a list of parcels, a *DAB* contains all the information needed to build a functional delivery.
If we add micro depots to the tours, we also have to consider the time it takes for the cargo bikes to move parcels from this depot to the bike. The trivial approach would be to just reuse the *DAB* class to build a “reverse” delivery, a pick-up. Using this method, we unfortunately run into problems quite early on: An instance of *DAB* verifies invariants after having parcels added to it, one of which is that all parcels in it must have the same delivery location. Pick-ups differ from this, as the parcels a cargo bike picks up at any

given micro depot do not necessarily have to go to the same destinations, in the vast majority of cases this assumption would be wrong. We therefore do not only have to forgo checking this invariant, but we must not do it at all.
To build a pick-up, we use a *PAB*[1], similar to the *DAB* explained above. A diagram of this implementation can be seen in figure 4.1.

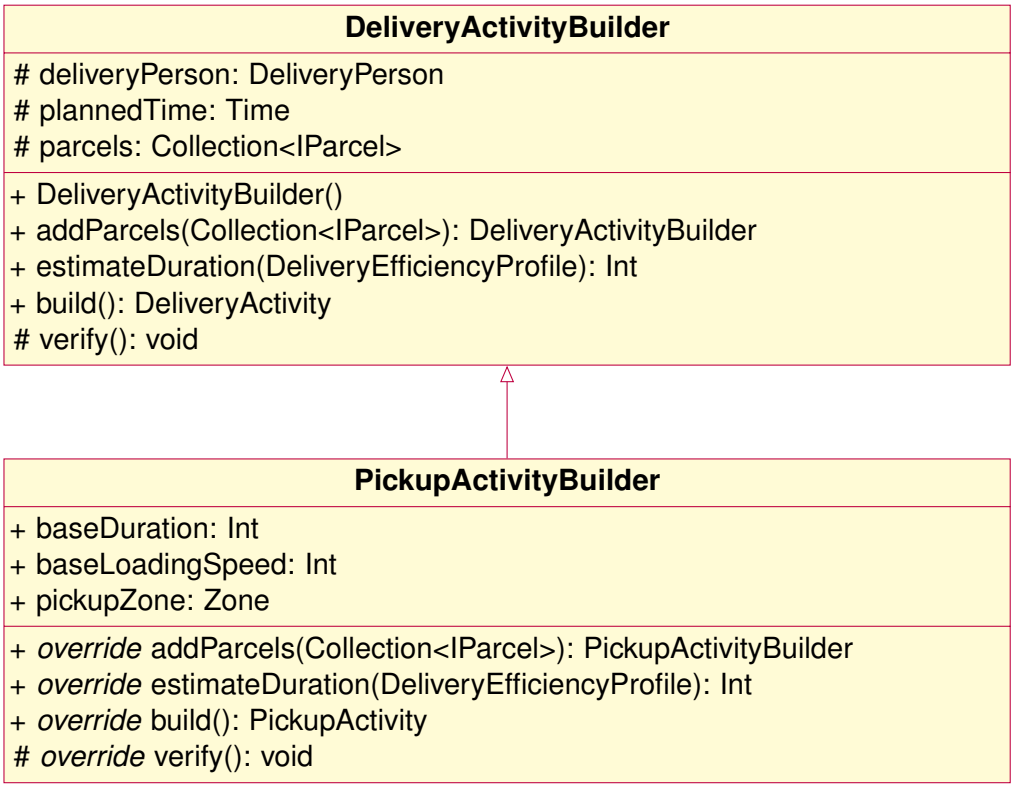


Figure 4.1: Relationship between deliveries and pickups

4.2 Micro depots

The first hurdle that the switch to cargo bikes presents is the vastly decreases storage space. To solve this issue we will implement micro depots, the concept of which have first been presented in Chapter 1.2, as well as in the beginning of this chapter. This allows for tours that are no longer limited by the amount of space a cargo bike has available. For the sake of simplicity we can assume, that a cargo bike can not only restock at a micro depot, but can also return the parcels, that could not be delivered.
A simple approach to micro depots is placing them at the exact location, where the cargo bike runs out of parcels. Algorithm 3 implements this procedure by extending

[1] PickupActivityBuilder

algorithm 1 found in Chapter 3.1. As in Algorithm 1 we assume all lists to be strictly ordered. The first addition are two new fields in lines 3 and 14.

pickups contains a running list of all planned pick-ups. Once all tours have been planned, *pickups* (line 3) can be used as additional output of *logiTopp* that contains all the details about the needed pick-up destinations and times for all cargo bikes.
The set *currentMicroDepotPickups* (line 14) contains all the parcels that have a planned delivery, but are not yet part of a planned pick-up. Those parcels will be included in the last pick-up that takes place, before the cargo bike reaches the parcel's location.
The new logic is introduced mostly in lines 16 to 34. We remember that our model assumes, that all parcels are identical, but the amount of parcels a cargo bike can carry is drawn from a normal distribution (compare chapter 2.1. We can make use of this fact by indexing all our deliveries to determine, at which point we run out of cargo space. At index 0 in line 17 we are at out first delivery. Since we started the tour at the delivery centre, the cargo bike is at maximum capacity. Placing a micro depot at the main depot is not needed, we can thus skip this index. Index *capacity* in line 19 is interesting, since at this point we have made *capacity* delivery attempts. Every parcel that has been part of the tour up until now has either been delivered, or will return to the delivery centre and be rescheduled for another delivery attempt. Unfortunately we have not yet determined, which parcels will be in this pick-up, but we know at which index the pick-up will take place. For now we will remember the time, zone and the index this pick-up will take place and finish it at a later time. We add a placeholder element in line 29 that we will later replace with the actual pick-up.
Line 24 covers all the remaining cases. Every time the current index is a multiple of the *capacity*, we arrive at a new pick-up location and can now finish the previous pick-up. To build a pick-up we need the *person* driving the cargo bike, information about the location (*lastPickupZone*) and time (*lastPickupTime*), as well as the parcels we have to load the bike with, which we have accumulated in *currentMicroDepotPickups* in line 44. Once we have built the pick-up we can update all required fields, so we have the required information once we arrive at the next pick-up.

Once we reach the end of the algorithm we have to consider the last micro depot of this tour. This has not yet been built since we only build a micro depot once we reach the next one has been reached. To address this issue we add lines 45ff.

We can notice that even though we update the time in line 32, the time taken for loading the cargo bike is only added to the time counter at the following pick-up location. This does not present an issue, since for now we are only making sure, that the total work we assign to the delivery person does not exceed their *remainingWorkTime* and the order of the tour is correct. The simulation does not care in which order we added the entries in *assignParcels*.

Algorithm 3 ZoneTSP.MicroDepots.assignParcels

```
1:  lastPlannedTime ← Time.start
2:  parcelTour ← {}
3:  pickups ← {}
4:  procedure ASSIGNPARCELS(deliveries, person, currentTime, remainingWorkTime)
5:      if parcelTour is outdated then
6:          parcelTour ← planZoneTour(person, deliveries, currentTime)
7:          lastPlannedTime ← currentTime
8:      end if
9:      assignedParcels ← {}
10:     capacity ← selectCapacity(person)
11:     lastZone ← person.distributionCenter.zone
12:     time ← currentTime
13:     endOfWork ← currentTime + remainingWorkTime
14:     currentMicroDepotPickups ← {}
15:     for all delivery ∈ parcelTour do
16:         index ← parcelTour.indexOf(delivery)
17:         if index = 0 then
18:             //
19:         else if index = capacity then
20:             pickupIndex ← assignedParcels.size
21:             assignedParcels.add(null)
22:             lastPickupTime ← time
23:             lastPickupZone ← lastZone
24:         else if index % capacity = 0 then
25:             currentPickup ← buildPickup(
                    person, lastPickupZone, lastPickupTime, currentMicroDepotPickups)
26:             pickups.add(currentPickup)
27:             assignedParcels.insertAt(pickupIndex, currentPickup)
28:             pickupIndex ← assignedParcels.size
29:             assignedParcels.add(null)
30:             lastPickupTime ← time
31:             lastPickupZone ← delivery.zone
32:             time ← time + pickupTime(currentMicroDepotPickups, person)
33:             currentMicroDepotPickups ← {}
34:         end if
```

Since the TSP is solved on the zone layer, and the order of the deliveries within a zone is not considered at all (compare chapters 3.1 and 3.2) a further possible step is to merge all micro depots within a zone into a single micro depot. This assumes that the micro depots are static and arrive before the first delivery person requires it. We will not present an implementation of this idea, but is easy to implement either in *logiTopp* itself or before evaluating the results using an external script.

ZoneTSP.MicroDepots.assignParcels

```
35:         if assignedParcels.size ≥ capacity then
36:             break
37:         end if
38:         tripDuration ← travelTime(person, delivery.zone, lastZone, time)
39:         deliveryDuration ← delivery.estimateDuration(person.efficiency)
40:         time ← time + tripDuration + deliveryDuration
41:         returnDuration ← travelTime(
                person, delivery.zone, person.distributioncCenter.zone, time)
42:         if time + returnDuration ≤ endOfWork then
43:             assignedParcels.add(delivery)
44:             currentMicroDepotPickups.add(delivery)
45:         else
46:             currentPickup ← buildPickup(
                    person, lastPickupZone, lastPickupTime, currentMicroDepotPickups)
47:             assignedParcels.insertAt(pickupIndex, currentPickup)
48:             pickups.add(currentPickup)
49:             break
50:         end if
51:         lastZone ← delivery.zone
52:     end for
53:     parcelTour.removeAll(assignedParcels)
54:     return assignedParcels
55: end procedure
```

4.3 Truck optimisation

In the previous section we have seen how we can add micro depots to cargo bike tours with a simple approach. While easy to implement, this specific approach might lead to a suboptimal distribution of pick-ups in “real world conditions”. While *logiTopp* has the exact coordinates of the delivery (and therefore pick-up) locations, it estimates the distance between any two locations based on a lookup table, that contains the average travel times between any two zones (compare Chapter 3.1). It can therefore happen that two locations that are next to each other, maybe on opposite sides of a main street, have a vastly inflated estimated distance, while the distance between nearly any location and a cul-de-sac in a small neighbourhood is under-estimated. This

also implies, that the distance between the delivery centre and any location in a zone is always estimated to be the same distance.
If we want to increase the quality of the pick-up locations, our algorithm should prefer locations that have a shorter distance to the delivery centre, for any provided metric. For this section to have a qualitative impact on the result, it is necessary to change the way the estimated distance is calculated. One possibility is to use the haversine formula [van Brummelen], which estimates the distance between two locations solely on their coordinates, using spherical distance. It is also possible to query a routing service for estimated travel times while building the TSP, but since a TSP requires a complete graph, the amount of queries would be quite high and a fitting data structure would be needed to cache the results for future simulations.

To find optimal pick-up locations we define a function that evaluates the quality of the location. This function receives the location as input and has has access to different metrics, such as the distance - in metres or minutes - to the delivery centre and the type of the road the location is located at, but many other factors are possible as well. We can define a threshold and plan a delivery centre on every location whose quality score is above this threshold. If the delivery bike is empty, we are forced to plan a delivery at this location. This approach requires context about the area we are delivering in, so that we can define we suitable threshold to neither over nor under assign pick-up locations. A refined approach is to - upon arriving at a pick-up location - look into the future and preview all possible locations that are part of the currently planned tour. This allows the use of an additional metric, the estimated load capacity of the cargo bike at any given delivery location, and only requires us to pick the location with the highest quality score within reachable distance, without having to define a threshold. This kind of evaluate function does not required previous information about the area we are delivering in. A possible implementation of this can be seen in Algorithm 4. The code snippet can be inserted into Algorithm 3 in line 17ff.

Algorithm 4 gets called once a pick-up location is reached. *deliveries* is a list that has size equal to the transport capacity of the cargo bike and contains the next planned deliveries of the giant TSP tour, that are planned after the current location for the bike. *evalFunction* is the function that evaluates the quality of the location. In this example its arguments are the location it has to evaluate, *nextLocation*, and the position in the list, represented be the current index of the location, *i* and the size of the list, *deliveries.size*. The *if-condition* in line 7 uses a greater equal to prefer locations, that occur later in the list to minimise pick-up locations.

While this approach might lead to an increase in pick-up locations compared to Chapter 4.2, it has the advantage, that the micro depots are more likely to be located at accessible locations, thereby possibly reducing the distance and time the trucks spend

Algorithm 4 findNextPickup

```
1: procedure FINDNEXTPICKUP(deliveries, evalFunction)
2:     bestIndex ← 0
3:     bestQuality ← 0
4:     for i ← 0, deliveries.size do
5:         nextLocation ← deliveries[i]
6:         quality ← evalFunction(nextLocation, i, deliveries.size)
7:         if quality ≥ bestQuality then
8:             bestQuality ← quality
9:             bestIndex ← i
10:        end if
11:    end for
12:    return bestIndex
13: end procedure
```

on the road.
As a final step we can solve a TSP on the pick-up locations to find the best route for the trucks delivering the micro depots.

4.4 Local TSPs

We have seen in previous sections (compare chapter 3.2), that the calculated tours contain quite a lot self intersections, which is undesirable in shortest paths, since shortest paths solutions do not tend to have these citation needed.
To work around this issue we could either move from solving zone TSPs to solving a much larger TSP instance on all delivery locations - this would vastly increase the amount of vertices and therefore edges and since TSP is NP-hard citation would result in large increase in time spent in planning the tours - or we keep the zone TSP but for each zone we solve a zone internal TSP, that only contains the locations within this zone.
Algorithm 5 describes how we can build such a local TSP. The idea is comparable to Algorithm 2, but we have to consider additional restraints. *start* and *end* denote at which locations we enter and leave the zone. We can find these locations by looking at the zone TSP and for any two neighbouring zones find the pair of locations that has the shortest distance. For this we have to check $zone_1.size * zone_2.size$ combinations, where the size of a zone is the amount of locations it has. We can optimize this by computing the convex hull of each set of locations, which has a worst-case complexity of $O(n \log n)$ [Graham und Frances Yao, 1983], n being the size of the zone. The expected size of a convex hull with random points drawn from a k sided polygon is in $O(k \log n)$ [Har-Peled, 2011], where n again is the size of the zone and k the amount of sides of the polygon

that represents the zone. Combining these results, we get an expected complexity of $O(k \log n * l \log m)$, which is an improvement over the original $O(n * m)$.

The first difference to Algorithm 2 occures in line 12. We identify the start vertex with the end vertex by adding an edge with weight 0 between the start and end point. In line 18 we use a TSP solver that forces the use of this 0 edge, a possible candidate is the greedy solver provided by the JGraphT library [Naveh, 2022]. If we remove this imaginary 0 edge we get a TSP “tour” that contains the shortest route from the start vertex to the end vertex.

Since the local TSPs are all solved individually a parallel implementation of this step is trivial to achieve by solving multiple TSP instances at the same time. Given the nature of the situation linear speedup can be expected.

Algorithm 5 localTSP

```
1:  procedure LOCALTSP(deliveries, start, end, currentTime)
2:      graph ← simpleWeightedGraph
3:      for all delivery ∈ deliveries do
4:          if delivery.location ∉ graph.vertices then
5:              graph.addVertex(delivery.location)
6:          end if
7:      end for
8:      for all location_1 ∈ graph.vertices do
9:          for all location_2 ∈ graph.vertices \ {location_1} do
10:             edge ← graph.addEdge(location_1, location_2)
11:             if location_1 = start && location_2 = end then
12:                 edge.setWeight(0)
13:             else
14:                 edge.setWeight(distance(location_1, location_2, currentTime))
15:             end if
16:         end for
17:     end for
18:     tspTour ← TspSolver.solve(graph)
19:     prefix ← {}
20:     suffix ← {}
21:     foundStart ← false
22:     for all location ∈ tspTour.vertices do
23:         if location = start then
24:             foundStart ← true
25:         end if
26:         for all delivery ∈ {delivery ∈ deliveries | delivery.location = location} do
27:             if foundStart then
28:                 prefix.add(delivery)
29:             else
30:                 suffix.add(delivery)
31:             end if
32:         end for
33:     end for
34:     return prefix + suffix
35: end procedure
```

5 Evaluation

Now that we have a good understanding of the framework (Chapter 2.2) and the additions made to it to accommodate the use of cargo bikes, let us look at a few results to evaluate how this change in method impacts the last-mile delivery of parcels.

5.1 Rastatt

In this evaluation we will consider the greater area around the German city Rastatt, close to Karlsruhe in Baden-Württemberg. Rastatt, which can be seen in Figure 5.1, is partitioned into 48 zones, the outskirts contain 16 more.

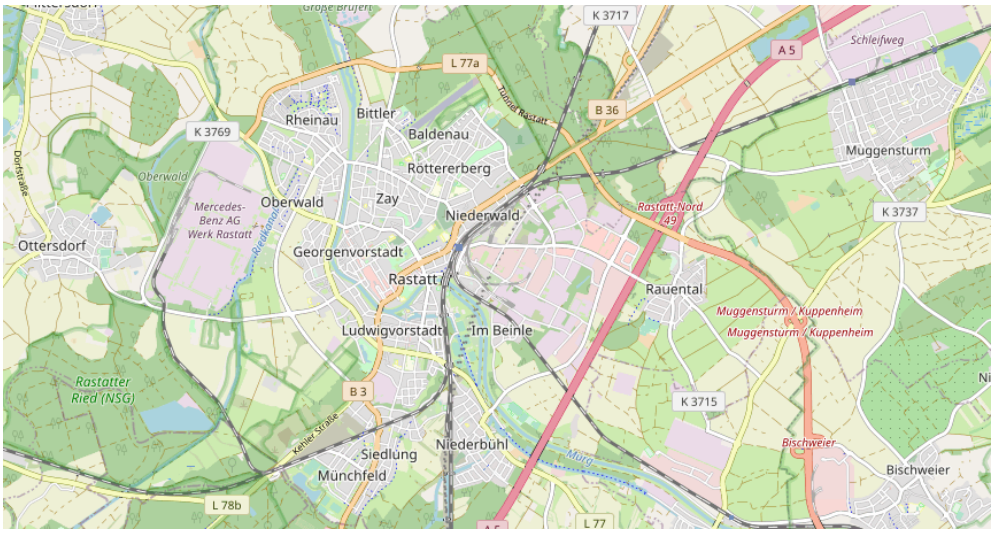


Figure 5.1: Rastatt with outskirts, from [OpenStreetMap, 2022]

As one can see in Figure 5.2 some zones are quite close to the centre of Rastatt, where the delivery centre will be located, whereas other zones are too far out to be a sensible destination for cargo bikes coming from Rastatt.

We will therefore evaluate one scenario where we only consider zones somewhat close to the delivery centre and one scenario, where we consider all zones as possible destinations.

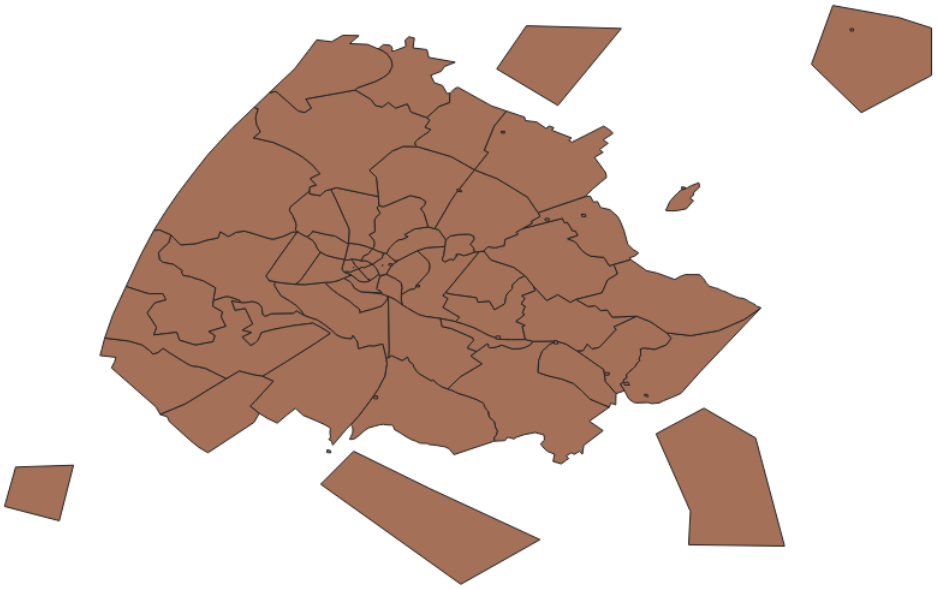


Figure 5.2: Some of the zones around Rastatt, data taken from [Kübler, 2021]

5.2 Of trucks and bikes

We have seen in Chapter 3.2, what a tour looks like if a truck is the vehicle of choice. If we replace the truck with a cargo bike but keep all other parameters the same, we can plan a simple tour for cargo bikes. The switch to cargo bikes is done by reducing the speed and cargo capacity of the simulated vehicle.
The result in this change can be seen in Table 5.1. This table contains the results for trucks with a mean of 160 and standard deviation of 16.0, and cargo bikes that share the same parameters. Only zones close to the Rastatt are considered, avoiding long travel times to zones in the outskirts. Furthermore we only consider a single main depot, located in the centre of Rastatt. It also contains the results of the code before the bug fixes in Chapter 3.3 were applied. These bugs had only a minor impact on the results. As described in Chapter 3.2 we can scale down the simulation, the results of which can also be seen in the table. At the first glance the total distance driven might appear high, but comparing it to the 1.5 million km the *Deutsche Post* drives daily [Post] the simulation seems to well within limits, if one accounts for the suboptimal way the deliveries are ordered and that the *Deutsche Post* is not the only delivery *"Courier, Express and Parcel" service providers* (CEPSP) in Germany. Table 5.2 compares this to a simulation that shares all the properties, but includes deliveries to all zones. One can clearly see, that the total distance travelled increases by $\sim$1000 km, for both truck 10% and bike 10%. The total travel time increases by $\sim$4 respectively $\sim$12 hours.

	total distance (km)	amount of drivers	average distance per driver (km)	average distance per driver per day (km)	average work time (h:m:s)	total work time (d:h:m:s)
Truck (1%) (bug)	1360.1	4	340.0	68.0	8:46:00	1:11:04:00
Truck (10%) (bug)	7423.3	25	296.6	59.3	9:56:51	10:08:41:15
Truck (100%) (bug)	62389.2	188	331.8	66.3	9:52:30	77:08:30:00
Bike (1%) (bug)	1363.1	5	272.6	54.5	5:59:18	1:05:56:30
Bike (10%) (bug)	8662.2	36	240.6	48.1	6:06:16	9:03:45:36
Bike (100%) (bug)	72732.0	344	211.4	42.2	6:38:31	95:04:49:44
Truck (1%) (fix)	1367.6	4	344.1	68.8	8:42:51	1:10:51:24
Truck (10%) (fix)	8059.3	27	298.4	59.6	9:28:41	10:15:54:27
Truck (100%) (fix)	68348.5	263	259.8	51.9	10:05:53	110:15:47:19
Bike (1%) (fix)	1411.8	4	352.9	70.5	8:06:45	1:08:27:00
Bike (10%) (fix)	8117.7	28	289.9	57.9	8:54:24	10:09:23:12
Bike (100%) (fix)	68620.6	272	252.2	50.4	9:46:46	110:20:00:32

Table 5.1: Comparison between trucks and cargo bikes, single depot, close zones only, average zone distance

	total distance (km)	amount of drivers	average distance per driver (km)	average distance per driver per day (km)	average work time (h:m:s)	total work time (d:h:m:s)
Truck (1%) (fix)	1648.2	4	412.0	82.4	8:46:54	1:11:07:36
Truck (10%) (fix)	9003.2	27	333.4	66.6	9:37:49	10:20:01:03
Bike (1%) (fix)	1637.4	4	409.3	81.8	7:59:10	1:07:56:40
Bike (10%) (fix)	9229.0	29	318.2	63.6	9:01:31	10:21:43:59

Table 5.2: Comparison between trucks and cargo bikes, single depot, all zones, average zone distance

Scaling	Total planned attempts (truck)	Total planned attempts (bike)
1%	282	265
10%	3.993	1.901
100%	38.077	16.699

Table 5.3: Amount of tried deliveries in Rastatt, close zones

During the course of the simulation multiple parcels will be delivered. Some of the deliveries will succeed during the first delivery attempt, some will fail multiple times. Table 5.3 list the total amount of tried deliveries. The amount of tried deliveries can vary greatly, since success of a single delivery is dependant on whether the recipient of the parcel is available at the time of delivery.

5.3 Micro depots & TSPs

Next we will look at what happens if we include micro depots and how different TSP solvers impact the quality of the tours. The 1% scaling option reduces the amount of parcels to such a small amount, that micro depots are not needed. We will therefore primarily look at results from the 10% scaling option, which introduces enough micro depot locations that results become noticeable.

Table 5.4 lists results from including micro depots using a selection of TSP solvers:

- Insert nearest neighbour
- Greedy
- 2 Approximation
- 3/2 Approximation

Further information about these algorithms can be found in the JGraphT documentation[1].

Table 5.5 uses the insert nearest neighbour solver to plan a complete tour of the close zones, this time using bigger cargo bikes than before. We can still see a slight improvement compared to the numbers in 5.1.

Bike (10%)	INN	Greedy	2 Approx	3/2 Approx
Total distance (km)	7883.9	7843.6	7867.9	7867.9
Amount of drivers	20	20	20	20
avg. distance / driver	394.1	392.1	393.3	393.3
avg. distance / driver / day	78.8	78.4	78.6	78.6
avg. work time	9:20:57	9:25:30	9:20:50	9:20:50
micro depots used	16	16	16	16

Table 5.4: Routing with micro depots, close zones, limited delivery agents

[1] https://jgrapht.org/javadoc/org.jgrapht.core/org/jgrapht/alg/tour/package-summary.html

Bike (100%)	INN
total distance (km)	68375.6
amount of drivers	263
avg. distance / driver	259.8
avg. distance / driver / day	51.9
avg. work time	10:08:40
micro depots used	5
total work time	111:03:59:20

Table 5.5: Routing with micro depots, close zones, limited delivery agents

6 Conclusion

In this thesis we explored the possibility of using cargo bikes instead of delivery trucks for last-mile parcel deliveries. To do this we looked at the challenges such a change introduces and the new restraints we have to consider. The most important distinctions this change introduces are the drop in cargo capacity and speed.
While we cannot change the speed of cargo bikes, the increased mobility is a somewhat natural counter to the first issue. To address the second issue we explored the concept of micro depots, small containers that can be placed along the tours of the cargo bikes, which vastly improves the effective delivery tour size of the bikes. This was first done by placing these depots at locations where the cargo bike would run out of parcels, but we later changed to a more sophisticated method that evaluated each location and placed micro depots based on the quality and the amount of parcels left on the cargo bike, which improved our results. The order of the pick-up locations can again be optimised using a TSP.
We also added a new option to estimated distance while precalculating the tours used by the cargo bikes using the delivery location's coordinates. While using the geographical distance based on the coordinates of the delivery locations is not a perfect estimate, it allows us to build and solve zone internal TSPs, which reduced the total length of the tour and comes closer to the tours used by real life CEPSPs. This addition increases runtime of the precalculation, but can be somewhat mitigated using parallelisation to solve multiple TSP instances at once.

6.1 Future work

We were not able to find a solution that satisfies all the challenges we faced, and even some of our successful approaches can be further optimised. This section fives a small overview for future improvements in finding optimal tours for cargo bikes.

Merging micro depots In our current estimation we determine the location of a micro depot without regards to other cargo bikes and information about the graph we are operation on. If we consider these factors when planning a tour it might be possible to reduce the amount of micro depots needed without achieving a worse result. Depending on the metric used this might also improve the quality of the results.

Considering new distance metrics For this thesis we have used two different metrics while calculating the tour of the cargo bikes. The first maps each location to its respective zone and uses a look up table to estimate the travel distance between the two zones. The second uses the haversine function the calculate the geographical distance, but does not include information about the actual travel time, which can differ a lot even for locations that are close to each other.
New metrics could include the distance, travel time or fuel consumption between locations, which would allow us to optimise our results according to our needs.

Building a TSP with dynamic edge weights The look up tables provide different travel times during different times of the day. Before a tour is planned one of these tables is chosen and used for the reminder of the current tour planning step. Using a temporal graph, a graph that may change as time progresses, tours that take into account different expected traffic situations during the day could be created. For these temporal TSPs several polynomial-time approximations exist, that could be used to calculated the tours [Michail und Spirakis, 2016].

Solving the vehicle routing problem The vehicle routing problem generalises the TSP. Instead of calculating a single giant tour, as we do in this thesis The vehicle routing problem considers a fleet of delivery vehicles operating from a given location and asks for the set of tours that minimises a given metric. As with the TSP, the vehicle routing problem is NP-hard [Toth und Vigo, 2002].

Bibliography

[Beasley 1983] BEASLEY, John E.: Route first—cluster second methods for vehicle routing. (1983)

[Briem und Kübler 2021] BRIEM, Lars ; KÜBLER, Jelle: *mobiTopp*. 2021. – URL https://github.com/kit-ifv/mobitopp

[van Brummelen] BRUMMELEN, Glen R. van: *Heavenly Mathematics: The Forgotten Art of Spherical Trigonometry*. – ISBN 9780691148922

[Ecke u. a. 2020] ECKE, L. ; CHLOND, B. ; MAGDOLEN, M. ; VORTISCH, P.: Alltagsmobilität und fahrleistung. (2020)

[Graham und Frances Yao 1983] GRAHAM, Ronald L. ; FRANCES YAO, F: Finding the convex hull of a simple polygon. In: *Journal of Algorithms* 4 (1983), Nr. 4, S. 324–331. – URL https://www.sciencedirect.com/science/article/pii/0196677483900135. – ISSN 0196-6774

[Har-Peled 2011] HAR-PELED, Sariel: On the Expected Complexity of Random Convex Hulls. In: *CoRR* abs/1111.5340 (2011). – URL http://arxiv.org/abs/1111.5340

[Korte 2008] KORTE: *The Traveling Salesman Problem*. S. 527–562. In: *Combinatorial Optimization: Theory and Algorithms*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2008. – URL https://doi.org/10.1007/978-3-540-71844-4_21. – ISBN 978-3-540-71844-4

[Kübler 2021] KÜBLER, Jelle: *mobitopp-rastatt*. 2021. – URL https://github.com/kit-ifv/mobitopp-rastatt

[Kübler und Reiffer 2022] KÜBLER, Jelle ; REIFFER, Anna: *logitopp*. 3 2022. – URL https://github.com/kit-ifv/logitopp

[Mallig u. a. 2013] MALLIG, N. ; KAGERBAUER, M. ; VORTISCH, P.: mobitopp–a modular agent-based travel demand modelling framework. (2013)

[Mallig und Vortisch 2017] MALLIG, N. ; VORTISCH, P.: Modeling travel demand over a period of one week: The mobitopp mode. (2017)

[Michail und Spirakis 2016] MICHAIL, Othon ; SPIRAKIS, Paul G.: Traveling salesman problems in temporal graphs. In: *Theoretical Computer Science* 634 (2016), S. 1–23. – URL https://www.sciencedirect.com/science/article/pii/S0304397516300342. – ISSN 0304-3975

[Naveh 2022] NAVEH, Barak: *JGraphT*. 2022. – URL https://jgrapht.org/

[OpenStreetMap 2022] OPENSTREETMAP: *Rastatt with outskirts*. https://www.openstreetmap.org. 2022

[Paul019 2021] PAUL019: *mobiTopp tour visualiser*. 2021. – URL https://paul019.github.io

[Post] POST, Deutsche *Über die Deutsche Post*. – URL https://www.deutschepost.de/de/u/ueber-uns.html. – Zugriffsdatum: 2022-04-15

[Reichel] REICHEL, Johannes: *City-Logistik: DHL kombiniert Lastenräder mit Klein-Containern und Mini-Hubs*. – URL https://logistra.de/news/nfz-fuhrpark-lagerlogistik-intralogistik-city-logistik-dhl-kombiniert-lastenraeder-mit-klein-containern-und-mini-hubs-13068.html. – Zugriffsdatum: 2022-04-14

[Reiffer u. a. 2021a] REIFFER, Anna ; KÜBLER, Jelle ; BRIEM, Lars ; KAGERBAUER, Martin ; VORTISCH, Peter: An integratedagent-based model of travel demand and package deliveries. (2021)

[Reiffer u. a. 2021b] REIFFER, Anna ; KÜBLER, Jelle ; BRIEM, Lars ; KAGERBAUER, Martin ; VORTISCH, Peter: Integrating urban last-mile package deliveries into an agent-based travel demand model. (2021)

[Statista] STATISTA: *e-commerce report 2019*

[Tischbeinahe 2021] TISCHBEINAHE: *DHL cargo bike in Vienna*. 2021. – URL https://de.wikipedia.org/wiki/Lastenfahrrad#/media/Datei:DHL_Cargo_Bike_in_Vienna_2.jpg. – Zugriffsdatum: 2022-04-13

[Toth und Vigo 2002] TOTH, P. ; VIGO, D.: *The vehicle routing problem*. 2002

[Union] UNION, Universal P.: *Data retrieved from Universal Postal Union database*.

– URL http://pls.upu.int/pls/ap/ssp_report.main2020?p_choice=AGGREG&p_language=AN.