TECHNISCHE UNIVERSITÄT WIEN
Vienna | Austria

D I P L O M A R B E I T

Solving the Consistent Vehicle Routing Problem based on Combinatorial Customer Grouping

Ausgeführt am Institut für

Stochastik- und Wirtschaftsmathematik der Technischen Universität Wien

im Rahmen des Studiums

Masterstudium Technische Mathematik, 066 394

unter der Anleitung von

Senior Lecturer Dipl.-Ing. Dr.techn. Josef Leopold Haunschmied

durch

Stefan Schwarzbach, BSc

Matrikelnummer: 01126221

Wien, am Unterschrift Unterschrift des Betreuers

Kurzfassung

In Zeiten, in denen das Erfüllen der Kunden-Bedürfnisse vergleichbare Wichtigkeit für Unternehmen im Paketzustellungssektor hat, wie die Kostenminimierung [16], gewinnt das Consistent Vehicle Routing Problem - erstmals erwähnt von Groër et al. [14] -, welches einen stabilen Lieferprozess, die Ankunftszeit und Kunden-Lieferanten Beziehung betreffend, im Fokus hat, zunehmend an Bedeutung im Operations Research Bereich. Diese Arbeit zeigt, dass das Gruppieren von Kunden in die eigens dafür definierte kombinatorische Struktur der Blöcke garantiert, dass die zwei Kernpunkte eines konsistenten Fahrzeug Routings erfüllt werden, nämlich Fahrer- und Ankunftszeitkonsistenz, während zusätzlich zur Erreichung von Kosteneffizienz auch die geographische Nähe der gruppierten Kunden berrücksichtigt wird. Gleichzeitig wird dadurch das größte Problem des ConVRPs, die Wechselwirkung der täglichen Routing Pläne unter einander, über die gesamte, betrachtete Zeit Spanne hinweg, gelöst, indem das Problem von der Dimension der Zeit losgelöst betrachtet wird und die Möglichkeit geboten wird Standard Lösungsalgorithmen, entwickelt für weniger komplexe VRPs, zu verwenden. Das macht das Konzept der Blöcke zu einem hoch flexiblen Werkzeug in der Berechnung von konsistenten Routen und motiviert zur fortführenden Anwendung.

Abstract

In times where customer satisfaction has similar importance to companies in package delivery commerce as cost minimization [16], the Consistent Vehicle Routing Problem - first mentioned by Groër et al. [14] - has gained attention in the operations research sector, focusing on a stable delivery progress in terms of delivery times and customer-deliverer relationship. This thesis shows that grouping customers in the specially defined combinatorial structure of blocks, guarantees satisfying the two key aspects of a consistent vehicle routing, i.e. driver and arrival time consistency, while additionally, cost efficiency is achieved by keeping the focus on the geographical neighbourhood of the partitioned customers. In addition, the greatest problem of the ConVRP, the interdependency of the daily routing plans over a time horizon of several days, is disarmed by considering the problem reduced by the dimension of time, giving the possibility of using standard solving strategies suggested for less complex types of Vehicle Routing Problems. This makes the concept of blocks a highly flexible tool in consistent route calculation motivating for further applications.

Contents

Introduction

In times where the customers and their satisfaction come to the fore of companie's interests [16], finding the shortest path to visit all customers is not the only target any more. Customer loyalty is one of the keywords of marketing. This Holy Grail can be reached if a social aspect is added to the delivery process [17]. Thinking about online shopping, the only interhuman contact is the package delivery between the driver of the package delivery company and the customer at the doorstep. First, this relationship is improved if the customer receives packages from the same driver [21]. Secondly, especially preferred by business customers (in B2B commerce) is a constant delivery time. Logistic operations are easier to manage if the delivery appears roundly at the same time [17]. These two aspects are summarized by the word consistency.
From the operational point of view these demands result in an optimization problem called the Consistent Vehicle Routing Problem (**ConVRP**) first mentioned in the context of the small package shipping industry by Groër et al. [14]. Driver and arrival time consistency give the common Capacitated Vehicle Routing Problem an additional dimension: the dimension of days of the considered time horizon. This especially can be seen in the exact problem formulation in form of linear mixed integer programs (MIP). The solution routes of every day effect the solution of every other day, because of the consistency aspects. This extension of the problem leads to a higher complexity and therefore to a smaller chance of finding an exact solution, respectively to a higher need of heuristical methods. Now, the usage of heuristics regarding the capacitated Vehicle Routing Problem is no innovation, so there is a lot of literature this thesis can recourse to, summarized in [25] and suggesting also different approaches to solve many variations of the standard Vehicle Routing Problem. While the main work on the Consistent Vehicle Routing Problem focuses on the use of common algorithms, this thesis will introduce possible adaptations of the state of art algorithms. Often the used heuristics lose their focus on consistency when it comes to the final computation of cost and time efficient routes.
The goal of this thesis is to keep all four objectives - time efficiency, cost minimization, driver-, and arrival time consistency - in the focus of the calclulation of the best delivery routes. This will be done by a highly dynamic approach combining customers with potentially different delivery days, but which are geographically like-positioned, to groups, called blocks, resulting in a simultaneous route computation over the predefined time horizon. The position of the customer in the routing plan will be proofed to play a sig-

nificant role regarding arrival time consistency, and blocks will be shown to be the ideal combinatorial construction to guarantee a stable routing position of a customer. Chapter 1 introduces the Consistent Vehicle Routing Problem, giving an overview on the state of the art works on this problem. The regarding mathematical modelling approaches, including exact formulations in form of mixed integer programs and differences to the strategy introduced in this thesis, are part of Chapter 2. The definition of the ConVRP treated in this work and its formulation as an MIP are described in the beginning of Chapter 3, followed by the main part of this thesis, the construction of customer blocks and the motivation behind this approach. Important combinatorial results are applied in the first section and can be looked up in Appendix A. At the end, Chapter 3 leads to the formulation of the ConVRP as a capacitated VRP on blocks, giving many possibilities of further solving strategies, which constitute the outlook in the conclusion (Chapter 5). The generation of instance data and the performance of the customer grouping in blocks are discussed in Chapter 4. For small instances, the previously mentioned VRP on blocks is solved, and the results are also part of this chapter.

Chapter 1

The Consistent Vehicle Routing Problem

In a competitive market, it might seem clear at first sight that focusing on minimizing costs while maximizing production efficiency is the best strategy to enhance profit. This strategy has changed in the last century. Keith [16] describes this recognitional progress and the shift from a profit driven company to a customer centred one as the "Marketing Revolution". This change has influenced the computation of routes in delivery companies, as well. The first time, the standard VRP was introduced by Dantzig and Ramser [5] as an extension of the Travelling Salesman Problem, called "Truck Dispatching Problem" [2], which simply focused on minimizing the travelled distance correlative to the costs. In the meantime, many enhancements of the VRP have arisen, like for example the Vehicle Routing Problem with Time Windows (VRPwTW) [7] and the Periodic Vehicle Routing Problem (PVRP) [4], including adaptations to constraints as well as to terms in the objective function, requiring a more customer orientated optimization.
The VRPTW allows the customer to choose the time the delivery arrives at the doorstep. An upper and a lower bound mark the time window in which the package should be delivered (for a more detailed description see [7]). This directly results in the necessity of tracing the time the deliverer arrives at the customer, briefly called *arrival time*, enlarging the corresponding mathematical model and also complicating heuristical solving approaches. Secondly, questions regarding the possibility of waiting times occur, giving the driver a degree of freedom to fulfil arrival time limitations, like for example the fulfilment of time windows.
These limitations are guaranteed by different strategies, which can basically be divided into two types. The strict or also called *hard constrained* way is formulating the observance of the time windows as an essential requirement for a solution of the VRPTW. The consequence is that it is more difficult finding feasible solutions, in some cases not even a single solution of the problem can be found. The *soft constrained* approach is penalising deviations from the time windows in the objective of the mathematical model. This means an influence on the optimality of a solution and may lead to inefficiency regarding the routes costs.[2]

Many results achieved for the VRPTW, like the formulation as a Linear Program or regarding the computation of solutions, are also applicable to the ConVRP. An even closer relatedness proceeds from the next type of Vehicle Routing Problem.
The paper of Christofides and Beasley [4] is one of the first mentioning the Periodic Vehicle Routing Problem namely. The PVRP is described as a Vehicle Routing Problem over a time horizon of several days, where the exact delivery days of each customer are not predefined. Every customer has its own delivery periodicity described for example in [4] as a set of feasible delivery day combination for every customer. In every set of combinations, a single combination is computed by a mathematical program giving the actual delivery days of a customer. Comparing this approach to the standard VRP, it can be seen that the customer has more freedom in choosing the days the delivery arrives, underlining the PVRP as a customer oriented extension of the VRP.
The Consistent Vehicle Routing Problem first introduced by Groër et al. [14] "is an extension of the Periodic Vehicle Routing Problem" ([18] p. 1) treating service requirements of a company of the small package shipping industry. Having "the customers to receive service from the same service provider at about the same time over multiple days" ([14] p. 630) is described as a "key component of high-quality customer service" ([14] p. 630). Like the PVRP also the ConVRP considers a Vehicle Routing Problem over a time horizon of several days. Every customer has its own set of delivery days, which can range from a single day to every day of the whole time horizon. One difference to the PVRP is, that these delivery days of each customer are known a priori, so the sets of delivery day combinations mentioned above and in [4] are singletons for each customer, meaning that every customer is able to choose the exact delivery dates. The second distinction characterising the ConVRP are the constrains and objective parameters guaranteeing consistency. So far, there are many different concepts of describing consistency mathematically (see Section 2.1 and the survey of Kovacs et al. [17]), but all have the interdependency between the daily routes in common, that the route of a single day highly influences the solution routes on the other days, due to the consistency requirements. This makes the Consistent Vehicle Routing Problem to one of the most complex and interesting variation of the VRP. In the meantime, highly diverse delivery industries are focusing on this type of routing strategy, reaching from the automobile spare part distribution to the pharmaceutical and health care sector [3]. Before giving a detailed mathematical formulation of the ConVRP the different types of consistency are discussed in the following section. Again I want to refer the reader to the survey of Kovacs et al. [17] which gives an interesting overview on works regarding consistency in Vehicle Routing Problems before 2014.

1.1 Consistency

Regarding the delivery of goods, consistency can basically identified with constancy. Changes in the delivery process attract the attention of the customer and is often associated with negative terms like unpunctuality and unreliability [21]. Conversely, constancy over several deliveries leads to a improved social relationship between customer

and deliverer and an easier time management of the customer [18]. Both are competitive advantages a company in the "highly competitive package delivery industry" ([14] p. 631) can provide their customers. But also in the companies point of view the higher customer loyalty is not the only advantage. Guaranteeing territory consistency to the driver is examined to result in the drivers familiarity with the routes and an increased performance by Zhong et al. [26]. Both the higher service level and constant arrival times (arrival time consistency) are summarized by the wording service consistency. The combination of arrival time consistency and driver consistency, i.e. every customer is supplied by the same driver, results in routes roughly remaining the same over the regarded time horizon. Effective routing algorithms tend to serve the customers of a cluster by the same vehicle, so a constant route results in a constant territory and therefore also in service consistency. In conclusion, it can be seen that arrival time consistency (AC) and driver consistency (DC) mentioned in the early work of Groër et al. [14] are already the two key aspects of fulfilling customer requirements.

1.1.1 Arrival time consistency

Similar to the VRPTW treating arrival time consistency and its measurement leads to the necessity of tracing the arrival times at a customer. Groër et al. [14] and following works of Subramanyam and Gounaris [24] and Kovacs et al. [18] suggest a hard constrained strategy, where the difference between two arrival times at a customer must not exceed a predefined bound. While other works, like Luo et al. [21], utilize a time window based formulation to describe a service consistent delivery, the approach of Groër et al. [14] can be interpreted as a dynamic time window for each customer. Important to mention is that the exact delivery time is not in customer's hand. Kovacs et al. [19] soften the hard constrained strategy by simply adding the maximum of all arrival time differences, to the objective, justifying this with the difficulty of predicting the previous mentioned bound if the "demand is fluctuating" ([17] p. 201). Going a step further, the arrival time consistency of every customer can be measured by an extra variable bound for each customer and considering the sum of all variable bounds to be minimized. A contrary and interesting idea of translating arrival time consistency in numbers is presented in the work of Feillet et al. [12]. Again a fixed time interval is given, but the difference of arrival times is allowed to exceed this limitations. If two arrival times of a single customer are set wider apart than the length of the interval, they must lie in different so called time classes. This approach results from the problem, described by the figures 1.1 and 1.2.
Measuring just the largest difference between two arrivals at a customers, leads to the problem that customer 1 and customer 2 have the same value of arrival time consistency (see Figure 1.1), even though customer 1 is served on four different times and customer 2 on just two different times. More intuitively the approach of measuring arrival time consistency with time classes in [12], values the arrival time consistency of customer 2 as the better one (see Figure 1.2). Similar to the length of a time class, the maximum number of time classes is also a priori limited. A soft constrained variant would be minimizing one of these constants. Important to mention is that a decrease of the maximal

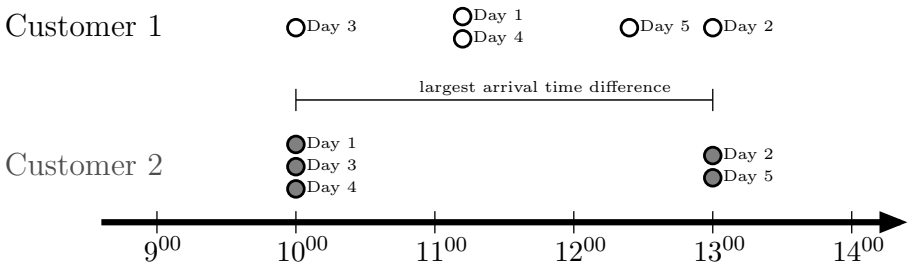


Figure 1.1: Arrival time consistency measured by the largest difference

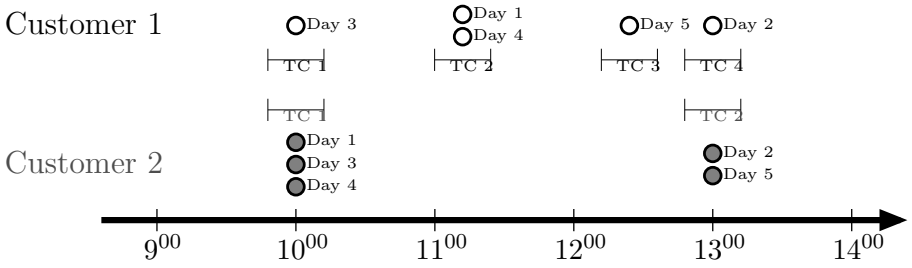


Figure 1.2: Arrival time consistency measured with time classes

time class length results in more time classes and reverse, so a multi-objective with both constants is redundant.

1.1.2 Driver consistency

As mentioned above driver consistency is the second important aspect of providing a consistent delivery service. Groër et al. [14] and later works of Kovacs et al. [18] and Campelo et al. [3] define driver consistency as the strict condition that a customer is only served by exactly one driver. This approach mostly leads to a hard solution finding process with possible cost intensive results. Limiting the number of drivers responsible for one customer by a predefined bound, instead of fixing it to 1, is suggested as a "more realistic" ([19] p.) assumption and is applicable in the works of Kovacs et al. [19] and Luo et al. [21]. The multi-objective version of the Consistent Vehicle Routing Problem in the work of Kovacs et al. [20] adds the maximal number of different drivers allowed to visit a customer to the objective and suggest a soft constrained approach. For this method it is necessary recording the exact number of drivers responsible for each customer. So it is again possible going a step further and value driver consistency as the sum of the exact recorded numbers of drivers responsible for each customer.

Chapter 2

Mathematical Modelling

Even if not described in detail by Dantzig and Ramser [5] the modelling of a VRP bases on a graph, symbolising the road network between the customers and one or more distinguished nodes, standing for the depots, where the vehicles have to leave before fulfilling their routes and return afterwards [25]. Usually a path in the graph from the start to the end depot corresponds with a route. As Section 3.2.1 shows, there are also other ways a route can be interpreted in the graph. The graph concept of modelling leads to the possibility of using graph specific algorithms or adaptions of them, like for example the Ford-Bellman algorithm in [10], to find cost minimal routes. The cost of a route typically result from the weights the arcs of the graph have, representing the travel time, distance or also energy demand [15] between two nodes. Especially in the case of a ConVRP, other costs, like a soft constrained arrival time or driver consistency, can occur. These costs do not necessarily depend on single routes any more, but the whole solution must be considered. The influence of the routing plan on one day on the route costs on another day is one of the biggest difficulties of these kind of VRPs. The following section summarizes different mathematical formulations of the ConVRP in form of linear mixed integer programs.

2.1 Mathematical formulations of the ConVRP

In most cases, the formulation of MIP serves the purpose of mathematically defining and describing the considered problem and not providing an exact solution strategy. Mathematically modelling the VRP uses in general boolean decision variables being related with the arcs of a mostly complete, directed graph [25]. So there is a squared dependence of the number of boolean variables from the number of vertices of the graph symbolizing the customers. Additionally, the capicitated VRP and therefore also the ConVRP are "known to be NP-hard (in the strong sense)" ([25]). Thinking of instances reflecting real world conditions the number of customers served by modern delivery companies results in a practical unability exactly solving these Mixed Integer Programs.
In summary, this section should just symbolize the differences of the mathematical problem definitions in literature to make it easier to compare the formulations of this thesis

with the state of the art works. In this sense, variable and constant names are adapted to give the reader a unified image of all approaches, whereby the used notation mainly orientates itself to the works of Toth and Vigo [25], Groër et al. [14], Feillet et al. [12], Kovacs et al. [18] and Campelo et al. [3]. With the best will it is tried to keep using calligraphic symbols for sets, capitals for parameters and small letters for variables. This unification is also used in the problem definition of this thesis in Chapter 3 and especially simplifies the comparison of the MIP parts guaranteeing driver and arrival time consistency (see Section 2.1.1). This following notation is at the best continued in the whole thesis.

Definition 2.1.1. Sets

$\mathcal{N}$ ... set of customers
$\mathcal{N}_0$... set of nodes including depots
$\mathcal{A}$ ... set of arcs
$\mathcal{D}$ ... set of days
$\mathcal{V}$ ... set of vehicles

[25] Assumed is a complete, directed graph $\mathcal{G} = (\mathcal{N}_0, \mathcal{A})$, where the set of nodes $\mathcal{N}_0 = \mathcal{N} \cup \{0, N+1\}$ includes the set of customers $\mathcal{N}$ and the depots 0 and $N+1$. For problems in which the departure and arrival depot are the same the depot $N+1$ is simply left out. Each arc in the set of arcs $\mathcal{A} \subseteq \mathcal{N}_0 \times \mathcal{N}_0$ represents the trip between two customers or between a customer and a depot. The mentioned time horizon, which gives the ConVRP it complexity, is described by Subramanyam and Gounaris [24] as a set of periods, but most work like for example [14], [12] and [19] define it more specific as a set of days $\mathcal{D}$. At last it is also necessary introducing the set of delivery vehicles $\mathcal{V}$.

Definition 2.1.2. Constants

N ... number of customers
D ... number of days
M ... a constant chosen sufficiently big
V ... maximal load volume of the vehicles
L ... maximal length of the arrival time interval
R ... maximal number of different drivers visiting a customer
T_{min} ... earliest time a driver may leave the depot
T_{max} ... latest time a driver must arrive at the depot

Modelling logical relations in form of linear constraints, often leads to the need of a sufficiently big constant M (see [9], [3]). Achieving a better performance in the optimization this constant should be set to the minimum of constants fulfilling the required conditions. The exact value of M varies from work to work, so it is not discussed in detail. In Section 1.1.1, different strategies are introduced to model arrival time consistency: hard and soft constrained formulations, largest difference and time class approaches. Regarding the context L may stand for the largest allowed difference between two arrivals at

a customer or also as the maximum size of the time class [12] and is sometimes also treated as a decision variable that is minimized in the objective. A single constant for the load capacity V of a vehicle assumes that the vehicle fleet is homogeneous. Working with a heterogeneous fleet, means that the fleet consists of different vehicle types and therefore the maximal load volume can vary between the vehicles. In the linear model, this would simply mean, that V has to be furnished with an index k referring to the regarding vehicle, which does not require any major changes in the problem formulation.

Definition 2.1.3. Indexes

i, j ... indexes referring to customers and depots
v ... index referring to a vehicle
d ... index referring to a day

Vertices and arcs are both furnished with weights, representing service time or demand at a customer and travel time, distance, energy demand or simply travel costs of an arc. The directedness of the graph means that the arc related weights may vary according to the direction, like it is the case in real world instances. The best route from one customer to another may not be the shortest or most efficient route, regarding time, energy or costs, if driven the opposite direction.

Definition 2.1.4. graph related parameters

$S_{i,d}$ ... service time at customer i on day d
$Q_{i,d}$ ... demand of customer i on day d
$P_{i,d}$ ... boolean constant representing if customer i requires service on day d
$T_{i,j}$ ... travel time between customer i and j

All Linear Programs describing the Consistent Vehicle Routing Problem are mixed integer programs, so the set of variables can be divided by their type, continuous, boolean or integer. Practically a boolean variable is an integer variable, which only attains the values 0 and 1. All variables, i.e. continuous, boolean and discrete, are always greater equal than zero. Conveniently these definitions replace the binary and non-negativity conditions in this chapter.

Definition 2.1.5. Variables

boolean:

$x_{i,j,v,d}$ ... representing if vehicle v drives from customer i to customer j on day d
$y_{i,v,d}$ ... representing if customer i is visited by vehicle v on day d
$z_{i,v}$ ... representing if customer i is visited by vehicle v on any day
u_v ... representing if vehicle v is used on any day

continuous:

$a_{i,d}$ ... arrival time at customer i on day d
$a^{\text{end}}_{v,d}$ ... arrival time of vehicle v at the arrival depot on day d
$a^{\text{start}}_{v,d}$ ... departure time of vehicle v at the departure depot on day d

As mentioned in the introduction of this chapter, the ConVRP was the first time defined by Groër et al. [14]. The following MIP is the regarding mathematical formulation with the renamed variables and constants defined above.

Mixed Integer Program 1. *[14]*
minimizing

$$\sum_{i\in\mathcal{N}_0}\sum_{j\in\mathcal{N}_0}\sum_{v\in\mathcal{V}}\sum_{d\in\mathcal{D}} T_{i,j}x_{i,j,v,d} \tag{2.1}$$

subject to

$$y_{0,v,d} = 1, \qquad \forall v\in\mathcal{V}, d\in\mathcal{D} \tag{2.2}$$

$$a_{0,d} = 0, \qquad \forall d\in\mathcal{D} \tag{2.3}$$

$$\sum_{v\in\mathcal{V}} y_{i,v,d} = P_{i,d}, \qquad \forall i\in\mathcal{N}, d\in\mathcal{D} \tag{2.4}$$

$$\sum_{i\in\mathcal{N}} y_{i,v,d}Q_{i,d} \le V, \qquad \forall v\in\mathcal{V}, d\in\mathcal{D} \tag{2.5}$$

$$\sum_{i\in\mathcal{N}_0\setminus\{j\}} x_{i,j,v,d} = \sum_{i\in\mathcal{N}_0\setminus\{j\}} x_{j,i,v,d} = y_{j,v,d}, \qquad \forall j\in\mathcal{N}_0, v\in\mathcal{V}, d\in\mathcal{D} \tag{2.6}$$

$$a_{i,d} + x_{i,j,v,d}(S_{i,d}+T_{i,j}) - (1-x_{i,j,v,d})M \le a_{j,d}, \qquad \forall i\in\mathcal{N}_0, j\in\mathcal{N}, v\in\mathcal{V}, d\in\mathcal{D} \tag{2.7}$$

$$a_{i,d} + x_{i,j,v,d}(S_{i,d}+T_{i,j}) + (1-x_{i,j,v,d})M \ge a_{j,d}, \qquad \forall i\in\mathcal{N}_0, j\in\mathcal{N}, v\in\mathcal{V}, d\in\mathcal{D} \tag{2.8}$$

$$\sum_{i\in\tilde{\mathcal{N}},j\in\tilde{\mathcal{N}}} x_{i,j,v,d} \le |\tilde{\mathcal{N}}| - 1, \qquad \forall\tilde{\mathcal{N}}\subseteq\mathcal{N} : |\tilde{\mathcal{N}}|\ge 2, \forall v\in\mathcal{V}, d\in\mathcal{D} \tag{2.9}$$

$$a_{i,d} + P_{i,d}(S_{i,d}+T_{i,0}) \le P_{i,d}T_{max}, \qquad \forall i\in\mathcal{N}, d\in\mathcal{D} \tag{2.10}$$

Driver consistency:

$$P_{i,d_1} + P_{i,d_2} - 2 \le y_{i,v,d_1} - y_{i,v,d_2} \le -(P_{i,d_1} + P_{i,d_2} - 2), \quad \forall i\in\mathcal{N}_0, v\in\mathcal{V}, d_1, d_2\in\mathcal{D} : d_1\ne d_2 \tag{2.11}$$

Arrival time consistency:

$$-L + M(P_{i,d_1} + P_{i,d_2} - 2) \le a_{i,d_1} - a_{i,d_2} \le L - M(P_{i,d_1} + P_{i,d_2} - 2), \quad \forall i\in\mathcal{N}_0, d_1, d_2\in\mathcal{D} : d_1\ne d_2 \tag{2.12}$$

The objective (2.1) shows that just the total travel time is minimized, which is said to be the "main cost driver" ([3] p. 134) in the delivery of goods. The inequations (2.2), (2.4) and (2.6) guarantee that all vehicles together leave the depot at the same time, supply all customers who require service on that day and return back to the depot.

Important to mention is that the y variables are auxiliary variables which can be fully replaced by the sum in (2.6). For the sake of clarity, these variables are kept in the MIP formulation (see also MIP 4). Constraint (2.5) shows that the ConVRP is an extension of a capacitated VRP ([14]) limited by the load volume of the vehicles. (2.3), (2.7), (2.8) and (2.10) are the constraints to monitor the arrival time at the customers and fulfilling that all vehicles leave the depot at time 0 and return to the depot not later than T_{max}. The combination of (2.7) and (2.8) prevent that the driver is allowed to wait to improve the arrival time consistency at a customer. Tracing the arrival times at a customer has the advantage that formulating subtour elimination constraints like (2.9), to eliminate subtours in the solution graphs, is not necessary.
Perfect driver consistency is guaranteed by (2.11), fulfilling $y_{i,v,d_1} = y_{i,v,d_2}$ if customer i requires service on both days d_1 and d_2. Constraint (2.12) is the hard constrained approach of providing arrival time consistency with a maximal difference of L. Here it is not necessary to formulate a Big M Constraint, because there are no restrictions for $a_{i,d}$ if customer i does not require service on day d. The solving algorithm is allowed to set $a_{i,d}$ to an arbitrary value within the surrounding of the other arrival times not greater than L. But before discussing the different consistency formulation approaches, basic differences are summarized.
In the meantime, the MIP above was improved by other works, in the sense of reducing constraints and variables, but also in the sense of adding constraints, called cuts, systematically shrinking the solution space. As mentioned above, Feillet et al. [12] leaves off the y variables, but additionally adds an index k to the arrival time variables $a_{i,d}$. The same adaption is done by Kovacs et al. [18] justified by an higher compatibility with their applications. It is also easy to see that changing (2.7) to

$$a_{i,d} + \sum_{v=0}^{V} x_{i,j,v,d}(S_{i,d} + T_{i,j}) - (1 - \sum_{v=0}^{V} x_{i,j,v,d})M \leq a_{j,d}$$

leads to the independence of the constraint from index v and therefore a significantly reduction of Big M Constraints. Comparatively bigger differences result from allowing a driver to wait with the goal enhancing arrival time consistency. Simply leaving off constraint (2.8) like it is done in Luo et al. [21] leads to the problem that waiting is not penalized in the objective. This leads to the importance of changing the kind of measuring the total service time of the driver. This is done by determining the times the vehicle leave the start depot, $a_{v,d}^{\text{start}}$, and the arrival times at the end depot, $a_{v,d}^{\text{end}}$, after fulfilling their routes. The difference $a_{v,d}^{\text{end}} - a_{v,d}^{\text{start}}$ then reflects the total time the vehicle is in service (see MIP 4).
I also want to refer the reader to the work of Subramanyam and Gounaris [24] formulating constraints for a branch and cut framework for the consistent travelling salesman problem, a single vehicle special case of the ConVRP.

2.1.1 Differences in the formulation of consistency

As discussed in the sections 1.1.1 and 1.1.2, the biggest differences in the mathematical formulation are the way to measure consistency and how to penalize inconsistency. Some approaches introduce inconsistency as costs in the objective, others simply does not allow inconsistent solutions. The first work of Groër et al. [14] formulates consistency as two constraints, i.e. (2.11) and (2.12), guaranteeing that each customer is served by the same driver and the arrival time difference between two visits is always lower than L.
Also the works of Kovacs et al. [18] and Campelo et al. [3] are suggesting a perfect driver consistency. Kovacs et al. [19] and Luo et al. [21] are introducing the variable $z_{i,v}$, which reflects if customer i is served by vehicle v on any day of the time horizon. Driver consistency is than guaranteed by the following constraints in a hard constrained way.

$$y_{i,v,d} \leq z_{i,v}, \qquad \forall i \in \mathcal{N}_0, v \in \mathcal{V}, d \in \mathcal{D} \tag{2.11a}$$

$$\sum_{v \in \mathcal{V}} z_{i,v} \leq R, \qquad \forall i \in \mathcal{N}_0 \tag{2.11b}$$

While both works ([19], [21]) present the additional variables $z_{i,v}$ as boolean, Constraint (2.11a) and the restriction by R guarantee that this implementation gives the possibility of defining $z_{i,v}$ as continuous variables. If no other limitations to $z_{i,v}$ exist the program will always try to choose the variables in a way to fulfil Constraint (2.11b). Controversially Constraint (2.11a) guarantees that if customer i is served by more than R different drivers, it is impossible for the program to choose the variables $z_{i,v}$ small enough to satisfy (2.11b). This is especially the same case when R is part of the objective, as in the work of Kovacs et al. [20] presenting a multi-objective version of the ConVRP and introducing a soft constrained variant of guaranteeing driver consistency.
A similar change of the constraints can be observed in the formulation of arrival time consistency. While Groër et al. [14] defines a strict maximal difference of arrival times at a customer of L (see also [18] and [24]), Kovacs et al. [19] and [20] introduce a soft constrained approach by adding L to the objective and slightly changing Constraint (2.12) to

$$(a_{i,d_1} - a_{i,d_2}) P_{i,d_1} P_{i,d_2} \leq L, \quad i \in \mathcal{N}, \forall d_1, d_2 \in \mathcal{D} : d_1 \neq d_2 \tag{2.12}$$

to get rid of the Big M constant.
As discussed in detail in Section 1.1.1, a completely different strategy to formulate arrival time consistency is presented by Feillet et al. [12] using the following notation.

Definition 2.1.6. Time classes

C ... maximal number of the time classes
$c_{i,d}$ ... variable indicating the time-class of the delivery of customer i on day d
f_{i,d_1,d_2} ... binary variable securing the consecutive numbering of the time classes

This work also manages formulating the logical condition $|a_{i,d_1} - a_{i,d_2}| > L \Rightarrow c_{i,d_1} \neq c_{i,d_2}$, that is two arrival times must lie in different time classes if their difference is higher than

L, in three linear constraints. The variable f_{i,d_1,d_2} equals 1 if the arrival time at customer i on day d_1 is earlier than the arrival time on day d_2 and their difference is larger than L. Constraint (2.12) is replaced by the following system of inequations.

$$a_{i,d_2} - a_{i,d_1} \leq L + M f_{i,d_1,d_2}, \qquad \forall i \in \mathcal{N}, \forall d_1, d_2 \in \mathcal{D} : d_1 \neq d_2 \tag{2.12a}$$

$$c_{i,d_1} + 1 \leq c_{i,d_2} + M(1 - f_{i,d_1,d_2}), \qquad \forall i \in \mathcal{N}, \forall d_1, d_2 \in \mathcal{D} : d_1 \neq d_2 \tag{2.12b}$$

$$c_{i,d} \leq C, \qquad \forall i \in \mathcal{N}, d \in \mathcal{D} \tag{2.12c}$$

[12] If the difference $a_{i,d_2} - a_{i,d_1}$ is bigger than L, the variable f_{i,d_1,d_2} attains the value 1 because of (2.12a). In this case Constraint (2.12b) fulfils that the time class number c_{i,d_1}, representing the time class of the arrival time at customer i on day d_1, is strictly lower than c_{i,d_2}. Consequently the following implication is guaranteed by the system of constraints above

$$|a_{i,d_2} - a_{i,d_1}| > L \Rightarrow c_{i,d_1} \neq c_{i,d_2}.$$

Additionally, the time classes are also ordered by time. The highest time class number is equal to the total number of time classes minus 1 and limited to C in (2.12c). Even though Feillet et al. [12] also formulate a hard constrained arrival time consistency, it is easily possible changing it to a soft constrained by minimizing C.

2.2 ConVRP solving methods

As other kinds of Vehicle Routing Problems, especially the ConVRP is confronted with the fact that computing solutions for great instances by simply typing the MIP in a solver is not possible, because of its NP-hard complexity [25]. So a wide variety of heuristical and exact solving approaches specialized on the ConVRP or similar problems have been introduced yet.

2.2.1 Template based heuristics

The first approach solving the ConVRP for great instances heuristically was also presented by Groër et al. [14], suggesting the use of a template solution. This is done by firstly ignoring customers, which require service on a single day and considering only the customers requiring service on multiple days, also called *frequent customers* ([18]), which are also the only customers influencing the consistency of a routing. The next step is calculating a template solution for frequent customers as if they all must be served on a single day. This idea follows from the policy described in [14], where the order of the customers served should remain the same on each day of the delivery time horizon, thus indirectly improve arrival time consistency. The advantage of calculating a template solution is that standard methods for solving a one-day capacitated VRP can be used. After the routing plan has been calculated for one day, all nodes representing customers, which do not require service on this day are deleted from the template route and all non-frequent customers supplied on that day are inserted in an heuristic way. This strategy

guarantees perfect driver consistency. Both works of Groër et al. [14] and Kovacs et al. [18] have to handle with the problems of defining an artificial vehicle capacity and maximal route length and fulfilling the hard constrained arrival time consistency. This is the case, because the template solution does not represent an actual single day route. Most of the times the template routes are significantly longer, so it is necessary to relax constants, like capacity and maximum driving time, to achieve cost efficient solutions. On the other hand a too soft relaxation results in an infeasible solution. This trade-off is hard to manage. The work of Kovacs et al. [19] names a second disadvantage of the template approach, the restriction of the search space and therefore a possible missing of better solutions. It also suggests the heuristic, presented in the following section, as the more flexible solving strategy.

2.2.2 Large Neighbourhood Search

The Large Neighbourhood Search (LNS) [23] is an heuristic trying to improve the current solution by destroying and repairing moves. Destroy operators remove nodes from the initial routes and repair operators insert these nodes on a better position [19]. Feillet et al. [12] heuristically calculates the most cost efficient routes for every day separately. The unification of those route plans builds the initial solution. Consistency is left out during this process. The way of measuring arrival time consistency (with time classes, see Section 1.1.1) leads to a special, simplified case of LNS solving. The day with the highest influence on arrival time consistency is chosen and all routes of this day are removed from the solution. Afterwards a VRP with multiple time windows is defined, regarding the customers time classes on the other days and with the goal of reducing the total amount of time classes. The solution of this VRPwMTW replaces the routing plan of the specific day. The disadvantage of this LNS approach, consisting of one destroy and one repair operator, is that it operates on single days. The routing plans of the untouched days highly influence the solution of the specific day, so only small steps in the search space are allowed.
A different strategy is presented by Kovacs et al. [19] using an LNS with five destroy and six repair operators, each focusing on a different part to optimize. The parts which are removed by the destroy operators are always whole customers, i.e. the nodes representing this customer on every day it requires service. The strategy considers the whole solution and does not focus on single days. The other difference is the choice of two initial solutions. The first one strongly emphasizes the total travel time, while arrival time consistency is just improved by adjusting the departure times of the vehicles from the depot. The second, very cost intensive solution is a route planning with perfect consistency by adding all frequent customers to different routes.

2.2.3 Exact methods

There were also several works using exact methods for solving routing problems with consistency considerations like the one of Feillet et al. [12] and Subramanyam and Gounaris [24]. The most common techniques are branch and price and branch and cut approaches.

Branch and Price

As discussed in the last section, Feillet et al. [12] uses an LNS based heuristic to improve arrival time consistency by destroying the whole routing plan of the day with the worst impact on arrival time consistency. With dependency on the remaining routes of the other days, a VRPwMTW is defined and solved with a branch and price based heuristic resulting from older works like [10] and [11]. The underlying branch and price algorithm is in detail described in Feillet [9]. The trick of this approach is the consideration of the following mathematical formulation instead common formulations of VRPs with worse linear relaxation ([9][sect. 3.1]).

Linear Program 2. *minimizing*

$$\sum_{r \in \mathcal{R}} T_r u_r$$

subject to

$$\sum_{r \in \mathcal{R}} Z_{i,r} u_r \geq 1, \ \forall i \in \mathcal{N}$$

The set $\mathcal{R}$ includes all possible feasible routes through the set of customers, where feasible is highly dependent from the type of VRP considered. Speaking of the VRPwMTW in [12] a feasible route would be a route fulfilling all vehicle and driver specific limitation and the constraints regarding the multiple time windows. Every route $r \in \mathcal{R}$ refers to an integer variable u_r, describing how often route r is used, and the total costs of the route T_r. The constant values $Z_{i,r}$ indicate if customer i is visited on route r, added up in the constraint it is guaranteed that every customer is visited at least once. The reason of not using an equality constraint and letting u_r be greater than 0 has computational and mathematical reasons. The reader is again referred to [9].
The catch of this formulation is that for greater problems finding the set $\mathcal{R}$ of all possible feasible routes is impossible because of the factorially increasing number of possibilities. The key is replacing $\mathcal{R}$ with a subset $\mathcal{R}_1 \subseteq \mathcal{R}$ and solving the linear relaxation of the formulation above called the restricted Master Problem. If the optimal routes are already contained in $\mathcal{R}_1$ then the optimal solution of the restricted Master Problem is also the optimal solution of the linear relaxation of the formulation above.
The next difficulty is finding optimal, new routes to add to $\mathcal{R}_1$. This step is called column generation, in which columns have to be found violating the constraint of the dual program of the restricted master problem:

$$\sum_{i \in \mathcal{N}} Z_{i,r} \tilde{u}_i \leq T_r = \sum_{(i,j) \in r} T_{i,j}, \quad \forall r \in \mathcal{R}_1$$

In this formulation, the variables $\tilde{u}_i$ are the nonnegative dual variables of the restricted master problem associated with each customer. The search for routes $r \in \mathcal{R} \backslash \mathcal{R}_1$ fulfilling $\sum_{i \in \mathcal{N}} Z_{i,r} \tilde{u}_i > T_r$ leads to an elementary shortest path problem with resource

constraints, which is solved by a label correcting algorithm ([10]) based on the work of Desrochers [8] and on the Bellman-Ford algorithm. Iteratively routes with negative reduced costs are added to the set $\mathcal{R}_1$ and the restricted MP is solved until no such improving route exist. This means that an optimal solution of the linear relaxation is found. If this solution is integral the algorithm stops. Otherwise a branching scheme has to be adapted and the process starts from new.
The formulation of an MIP where the routes are known a priori gives a high flexibility. Restrictions to the routes which would lead to difficult linear formulation can be left out algorithm on VRPs with consistency requirements is faced with problems like the multidimensionality and the fact that consistency can not be measured considering only a single route. The first problem may be solved by substituting routes with vehicles and their routing plan over several days or simply grouping deliveries of different days to blocks, like it is the approach of this thesis. Still it is not easy determining the value of consistency if a perfect driver consistency is not assumed and a customer may be visited from two different drivers. Defining the costs of driver consistency as $\frac{D_i - D_{i,r}}{D_i}$, where D_i is the number of days customer i requires service and $D_{i,r}$ are the number of stops at customer i on routing plan r, would solve the problem because of the fact that

$$\sum_{r:Z_{i,r}=1} \frac{D_i - D_{i,r}}{D_i} = |\{r : Z_{i,r} = 1\}|\frac{D_i}{D_i} - \frac{1}{D_i} \sum_{r:Z_{i,r}=1} D_{i,r} = |\{r : Z_{i,r} = 1\}| - 1.$$

Unfortunately the costs of arrival time consistency and also the feasibility of a route regarding arrival time consistency can only be determined if all stops of customer i are part of the same routing plan.
An other key issue regarding the use of the branch and price algorithm for the ConVRP is the computational aspect. The label correcting algorithm of [10] for solving the elementary shortest path problem with resource constraints reaches an acceptable runtime if the number of labels at each node is kept small. Leaving the restrictions of time windows multiplies the number of allowed labels at each node and slows down the whole algorithm.
Summarizing the works of Feillet et al. [10], Feillet et al. [11] and Feillet [9] give a great overview about the functionality of the branch and price algorithm and its application for the VRP with time windows and [12] shows one of the few possibilities using this algorithm for solving the ConVRP.

Branch and Cut

An other important and current work tackling a consistent routing with an exact algorithm is the one of Subramanyam and Gounaris [24]. The considered problem is the consistent travelling salesman problem solved by branch and cut algorithms based on three different but equivalent linear mixed integer formulations of the problem. The formulation clearly outperforming the other two - "in terms of number of instances solved, quality of lower bounds obtained and solution times" ([24] p. 394) - uses the Subtour

Elimination Constraints (SEC) of Dantzig et al. [6] and gets along without tracing the arrival time at every customer by introducing so called Inconsistent Path Elimination Constraints (IPEC).

Mixed Integer Program 3. *minimizing*

$$\sum_{d\in\mathcal{D}}\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}} T_{i,j}x_{i,j,d} \tag{2.1}$$

subject to

$$\sum_{i\in\mathcal{N}_0, i\neq j} x_{i,j,d} = \sum_{i\in\mathcal{N}, i\neq j} x_{j,i,d} = 1, \qquad \forall j\in\mathcal{N}_0, d\in\mathcal{D} \tag{2.2}$$

$$\sum_{i\in\tilde{\mathcal{N}}}\sum_{j\in\tilde{\mathcal{N}}} x_{i,j,d} \leq |\tilde{\mathcal{N}}| - 1, \qquad \forall\tilde{\mathcal{N}}\subsetneq\mathcal{N} : |\tilde{\mathcal{N}}|\geq 2, d\in\mathcal{D} \tag{2.3}$$

$$\sum_{k=1}^{s-1} x_{\gamma_k,\gamma_{k+1},d_\Gamma} + \sum_{k=1}^{t-1} x_{\delta_k,\delta_{k+1},d_\Delta} \leq |\Gamma| + |\Delta| - 1, \qquad \forall(\Gamma,\Delta)\in\mathcal{P}^{incons} \tag{2.4}$$

The number of SEC and the size of the set of all inconsistent pair of paths $\mathcal{P}^{incons}$ are increasing exponentially with the number of customers, so these constraints are added dynamically as cutting planes. A pair of paths (Γ,Δ) with $\Gamma = (\gamma_1,...,\gamma_s)_{d_\Gamma}$ and $\Delta = (\delta_1,...,\delta_t)_{d_\Delta}$ is defined as inconsistent if there exist indexes $i\in 1,...,s$ and $j\in 1,...,t$ such that $\gamma_i = \delta_j$, $d_\Gamma\neq d_\Delta$ and $|a_{\gamma_i,d_\Gamma} - a_{\delta_j,d_\Delta}| = |a_{\gamma_i,d_\Gamma} - a_{\gamma_i,d_\Delta}| > L$.
A stronger version of Constraint (2.4), called Tournament Constraint (2.5), is said to be achieving computational better results. With (Γ,Δ) it may also be the case that the pairs (Γ',Δ), (Γ,Δ') and (Γ',Δ'), where Γ', Δ' are the reversing paths of Γ, Δ, are inconsistent. Than Constraint (2.4) is replaced by Inequality (2.6).

$$\sum_{k=1}^{s-1}\sum_{l=k+1}^{s} x_{\gamma_k,\gamma_l,d} + \sum_{k=1}^{t-1}\sum_{l=k+1}^{t} x_{\delta_k,\delta_l,d} \leq s+t-3, \qquad \forall(\Gamma,\Delta)\in\mathcal{P}^{incons} \tag{2.5}$$

$$\sum_{k=1}^{s-1}(x_{\gamma_k,\gamma_{k+1},d} + x_{\gamma_{k+1},\gamma_k,d}) + \sum_{k=1}^{t-1}(x_{\delta_k,\delta_{k+1},d} + x_{\delta_{k+1},\delta_k,d}) \leq s+t-3, \qquad \forall(\Gamma,\Delta)\in\mathcal{P}^{incons} \tag{2.6}$$

It is also sufficient considering only nondominated pairs of paths, where the relation of dominance is simply the inclusion relation.
Given a solution defining new IPEC cuts is done in two steps. At first all paths with an arc (i,j) with $x_{i,j,d} > 0.5$ are determined, which can be done in polynomial time because Constraint (2.2) guarantees that at most one $x_{i,j,d} > 0.5$ exists for every node i. Secondly every possible pair of customers γ_1, γ_s and every pair of paths including both of them is checked to be inconsistent by proofing that one of the following conditions

is complied. The paths $\Gamma = (\gamma_1, ..., \gamma_s)_{d_\Gamma}$ and $\Delta = (\delta_1, ..., \delta_t)_{d_\Delta}$ are defined as above, $d_\Gamma \neq d_\Delta$ is assumed and $\tau(\Gamma)$ is defined as the total travelling time of path Γ.

$$\gamma_1 = \delta_1 = 0 \wedge \gamma_s = \delta_t \wedge |\tau(\Gamma) - \tau(\Delta)| > L$$
$$\gamma_1 = \delta_1 \neq 0 \wedge \gamma_s = \delta_t \neq 0 \wedge |\tau(\Gamma) - \tau(\Delta)| > 2L$$
$$\gamma_1 = \delta_t \neq 0 \wedge \gamma_s = \delta_1 \neq 0 \wedge |\tau(\Gamma) - \tau(\Delta)| > 2L$$

If no pair fulfilling one of these statements can be found the solution is consistent. If it is not integral, branching is applied.
The approach of Subramanyam and Gounaris [24] works well for a hard constrained arrival time consistency, but is not applicable if arrival time differences should just be penalized in the objective function. Developing the consistent travelling salesman problem to a ConVRP also leaves the question of how to treat with driver consistency, unanswered.

2.2.4 Comparison to this approach

The solving approach of this thesis intensively orientates itself by works introduced in the last sections and also mentioned in the following. The goal is a mathematical heuristic derived from the motivation behind template approaches introduced in [14] and [18], i.e. reducing the ConVRP to a single day VRP and keeping the order of customers, but also guaranteeing flexibility in the examination of search space like in [19] and [20].
The key idea is that not only a constant order of the customers is important for arrival time consistency, but also a nearly constant position of the customer in the routing plan. Developing this thought further results in a node grouping strategy, also mentioned in the work of Campelo et al. [3] but implemented in a more combinatorial way, which is presented Section 3.1. These set of customers are ideally of the form that within a single set the same amount of customers have to be served on every day. Also the distances between the customers in a group are chosen as small as possible. The creation of this groups, which will be called (complete, indivisible) blocks, is a combinatorial problem distantly related to Tetris. Analysing this combinatorial structure leads to interesting results, which are used for the computation of blocks (see Appendix A).
The direct conclusion of this approach is the possibility of defining a VRP on blocks which is totally independent of the amount of days. Blocks are treated as single nodes, which have its own service time, travel time and load. These values can be defined as vector-valued, making it redundant estimating a new capacity and time limit for the block routes. Every vehicle has its own list of blocks, which can be directly translated in the daily routing plan, resulting in a consistent occupancy rate of the deliverer. This saves costs and is also in the interest of the drivers, raising their satisfaction. Every customer is part of a single block, guaranteeing a perfect driver consistency and also the order of the customers is kept this way. In summary, this approach has all the advantages of a template solution. Additionally, it is possible to disregard the difficulty strongly related to the work with template solutions, of relaxing vehicle specific limits to get optimal routes. The use of blocks has the advantage that the actual vehicle limits

can be considered and gets along without a complicated prediction of new bounds as mentioned in [19].
Thinking about the greater flexibility of the LNS heuristic, it is possible to introduce a block based LNS. Beside adapting standard operators for blocks, also completely new operators can be defined. Block splitting destroy operators would increase the flexibility and the amount of possible moves in the search space. Defining consistency as soft constrained conditions just changes the way a solution is valued, but also implementing hard constrained consistency is an option by simply restricting the operators. For example when a strict driver consistency of one driver per customers is asked, it is not possible for a operator to split a customer into two blocks.
Grouping customers into blocks can also be considered as a simple preprocessing technique. An increasing amount of customers, which are supplied on every day of the time horizon, significantly simplifies the problem. A block may then be treated as a single customer requiring service on every day.
This thesis should show that the concept of blocks raises the ConVRP on a different level and also motivate for further usage. As a meta heuristic, the combinatorial structure of blocks can be used to adapt many existing heuristic approaches to make them applicable for Vehicle Routing Problems with consistency considerations.

Chapter 3

Solution Procedure

As described in Chapter 1 and 2, there are several variants in the formulation of consistency for VRPs. Giving the reader a detailed description of what sort of Consistent Vehicle Routing Problem is treated in this thesis, the main properties are listed, followed by an exact formulation as a Mixed Integer Program. The following conditions are assumed:

- homogeneous vehicle fleet
- drivers are allowed to wait
- flexible departure time from the depot
- vehicle amount and consistency are soft constrained

A homogeneous fleet of sufficiently enough vehicles with a maximal load volume of V is assumed for the delivery. In the sake of improving arrival time consistency, the drivers are allowed to wait before visiting the next customer and also the time of departure from the depot and arrival at the depot can be chosen arbitrarily. Additionally, all arrival times must lie within the interval $[T_{min}, T_{max}]$. The amount of drivers responsible for a customer and also the time difference between two arrivals at a customer is not bounded. Also the number of vehicles is not limited, because of the fact that a higher amount of vehicles has a positive effect on consistency. Coincidentally, an increase of the used vehicles and allowing waiting times, result in higher vehicle and time specific costs. This trade off is best described in a multi objective MIP. The objective function includes the total travel time TT - measured as the time differences between departure and arrival at the depot, therefore including the pricing of waiting times - and the amount of vehicles VC. Driver Consistency DC is measured as the amount of drivers responsible for each customer and arrival time consistency AC as the maximal difference between two arrivals. The different costs are proportioned by the weights α, β and γ. For additional constant and variable definitions the reader is referred to Section 2.1. The following MIP is derived from the works of Groër et al. [14] and Kovacs et al. [19].

Mixed Integer Program 4. *minimizing*

$$TT + \alpha VC + \beta DC + \gamma AC \tag{3.1}$$

with

$$TT := \sum_{d \in \mathcal{D}} \sum_{v \in \mathcal{V}} a_{v,d}^{end} - a_{v,d}^{start} \tag{3.2}$$

$$VC := \sum_{v \in \mathcal{V}} u_v \tag{3.3}$$

$$DC := \sum_{i \in \mathcal{N}} \sum_{v \in \mathcal{V}} z_{i,v} \tag{3.4}$$

$$AC := \sum_{i \in \mathcal{N}} l_i \tag{3.5}$$

subject to

$$y_{0,v,d} = y_{N+1,v,d} = 1, \qquad \forall v \in \mathcal{V}, d \in \mathcal{D} \tag{3.6}$$

$$\sum_{v \in \mathcal{V}} y_{i,v,d} = P_{i,d}, \qquad \forall i \in \mathcal{N}, d \in \mathcal{D} \tag{3.7}$$

$$\sum_{i \in \mathcal{N}} y_{i,v,d} Q_{i,d} \leq V, \qquad \forall v \in \mathcal{V}, d \in \mathcal{D} \tag{3.8}$$

$$\sum_{i \in \mathcal{N}_0 \setminus \{j\}} x_{i,j,v,d} = \sum_{i \in \mathcal{N}_0 \setminus \{j\}} x_{j,i,v,d} = y_{j,v,d}, \qquad \forall j \in \mathcal{N}_0, v \in \mathcal{V}, d \in \mathcal{D} \tag{3.9}$$

$$\sum_{d \in \mathcal{D}} \sum_{j \in \mathcal{N}} x_{0,j,v,d} \leq u_v, \qquad \forall v \in \mathcal{V} \tag{3.10}$$

$$a_{i,d} + S_{i,d} + T_{i,j} - (1 - \sum_{v \in \mathcal{V}} x_{i,j,v,d}) M \leq a_{j,d}, \qquad \forall i \in \mathcal{N}, j \in \mathcal{N}, d \in \mathcal{D} \tag{3.11}$$

$$a_{i,d} + S_{i,d} + T_{i,N+1} - (1 - y_{i,v,d}) M \leq a_{v,d}^{end}, \qquad \forall i \in \mathcal{N}, v \in \mathcal{V}, d \in \mathcal{D} \tag{3.12}$$

$$a_{i,d} - T_{0,i} - (1 - y_{i,v,d}) M \geq a_{v,d}^{start}, \qquad \forall i \in \mathcal{N}, v \in \mathcal{V}, d \in \mathcal{D} \tag{3.13}$$

Driver consistency:

$$y_{i,v,d} \leq z_{i,v}, \qquad \forall i \in \mathcal{N}, v \in \mathcal{V}, d \in \mathcal{D} \tag{3.14}$$

Arrival time consistency:

$$(a_{i,d_2} - a_{i,d_1}) P_{i,d_1} P_{i,d_2} \leq l_i, \quad \forall i \in \mathcal{N}, d_1, d_2 \in \mathcal{D} : \ d_1 \neq d_2 \tag{3.15}$$

Boolean, integer and non-negativity constraints

$$x_{i,j,v,d},\ y_{i,v,d} \in \{0,1\}, \qquad \forall i \in \mathcal{N}_0, j \in \mathcal{N}_0, v \in \mathcal{V}, d \in \mathcal{D} \tag{3.16}$$

$$a_{i,d},\ a_{v,d}^{start},\ a_{v,d}^{end},\ l_i \in [T_{min}, T_{max}], \qquad \forall i \in \mathcal{N}, v \in \mathcal{V}, d \in \mathcal{D} \tag{3.17}$$

$$z_{i,v},\ u_v \in [0,1], \qquad \forall i \in \mathcal{N}, v \in \mathcal{V} \tag{3.18}$$

3.1 Construction heuristic

The construction heuristic used for the generation of initial solutions in this thesis results from the idea that if the customer's position on the daily routing plan of the delivering vehicle remains roughly constant, then also the arrival time of the vehicle at the customer vary less. So the first step in constructing an initial solution is ignoring travel and service times and focusing only on the position of the customers in the delivery routes. This means that arrival time consistency is in the foreground of this construction heuristic. Additionally, the driver consistency will be perfect, because of the fact that always all stops of a customer are visited by a single vehicle. In many details of the construction algorithm (see Section 3.1.3 and Section 3.1.4), we will see that minimizing costs regarding the travel time and the amount of vehicles is not fully ignored and is also an important part of this heuristic. The following sections should give an image, to get a deeper knowledge of the core idea of the algorithm.

3.1.1 Glass column heuristic

As discussed above, the order in which the customers are served is the only important information at first. The problem of getting the best order, which ensures that every customer will be served on nearly the same position on each day, is translated in a more imaginable problem of throwing coloured balls in glass columns. Every stop is visualized by a coloured or marked ball, where the colour or number represents the customer, which is supplied at this stop. The glass columns represent the routes of a vehicle.
In detail, if we consider a time horizon of D days, then each vehicles delivery plan is decoded in D glass columns and the balls in them. The lowest ball representing a stop at customer i in column d means, that customer i is the first customer to be served on day d by the vehicle. The customer which is represented by the ball lying above the first one in the glass column is the next to be supplied by the vehicle on the day d and so on. Getting an easier formulation, the interchangeability between the terms *ball* and *stop* and between *column* and *day* will be used to formulate abbreviations like *adding the stop i to the column d*. It is also necessary to define what is meant by adding a whole customer to a vehicle. Every customer has a stop or a ball on each day the customer require service. So if *customer i is added to the glass columns of vehicle v*, all balls representing the stops of a customer i are thrown in the glass column representing the days the customer requires service on. For example, as it is shown in Figure 3.1, customer 1 is served on Monday, Tuesday and Thursday.
The next step is finding the best way of throwing the balls in the columns, where best does not only mean that every stop of a customer lies on a minimal number of rows. Also the distance between the rows is important. Again Figure 3.1 shows that customer 3 on the left and customer 3 on the right side lie on the same number of rows, but the distance between the rows and therefore the arrival time difference of customer 3 is bigger in the left graphic.
For a better formulation of the problem above, it is necessary to define the level of the glass columns.

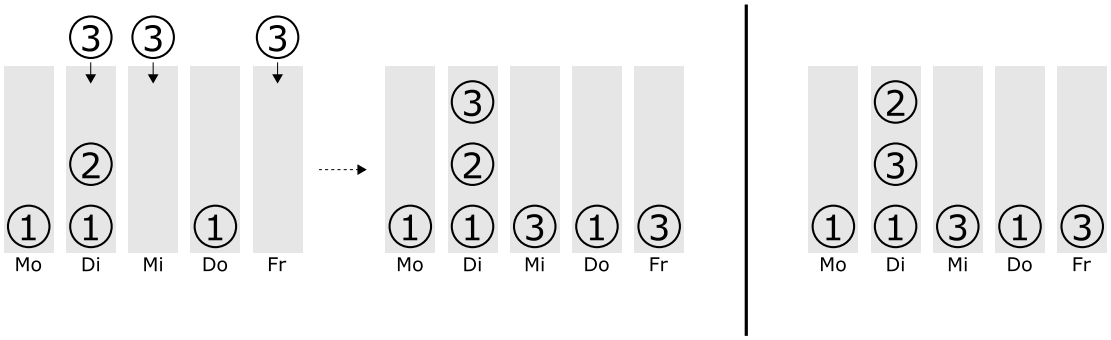


Figure 3.1: Glass Column Heuristic

Definition 3.1.1. Assuming D glass columns and a subset of customers $\{i_1, ..., i_m\}$, we will call the number of balls in column d after adding the k-th customer $\lambda_{k,d} = \sum_{k_0=1}^{k} P_{i_{k_0},d}$. Then the k-th *level* λ_k, the *difference* of a level Δ_k and the customer specific difference δ_k are defined as

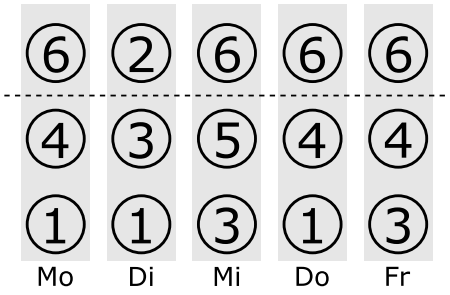
$$\lambda_k := (\lambda_{k,1}, ..., \lambda_{k,D}) \tag{3.19}$$

$$\Delta_k := \max\{\lambda_{k,d_1} - \lambda_{k,d_2} \mid d_1, d_2 \in \{1, ..., D\}\} \tag{3.20}$$

$$\delta_k := \max\{P_{i_k,d_1} P_{i_k,d_2} (\lambda_{k,d_1} - \lambda_{k,d_2}) \mid d_1, d_2 \in \{1, ..., D\}\} \tag{3.21}$$

A level λ_k is called *null level* if its difference $\Delta_k = 0$.

Figure 3.2 and Table 3.1 should give a context between glass columns and the definitions above.

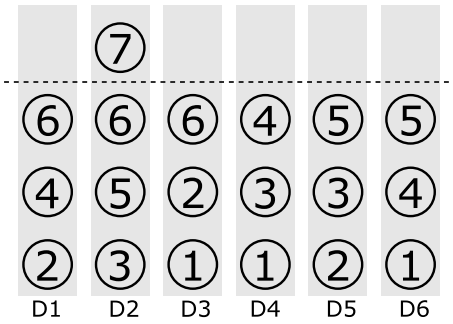


Cust.	Level	Differences	
2	$l_6 = (3,3,3,3,3)$	$\Delta_6 = 0$	$\delta_6 = 0$
6	$l_5 = (3,2,3,3,3)$	$\Delta_5 = 1$	$\delta_5 = 0$
5	$l_4 = (2,2,2,2,2)$	$\Delta_4 = 0$	$\delta_4 = 0$
4	$l_3 = (2,2,1,2,2)$	$\Delta_3 = 1$	$\delta_3 = 0$
3	$l_2 = (1,2,1,1,1)$	$\Delta_2 = 1$	$\delta_2 = 1$
1	$l_1 = (1,1,0,1,0)$	$\Delta_1 = 1$	$\delta_1 = 0$

Figure 3.2 & Table 3.1: Glass Column Heuristic and level definitions in relation

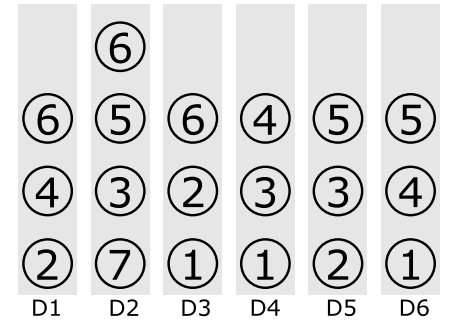
If we now search for a solution, where every customer lies on a minimal number of rows which in turn lie as close as possible to each other, it is possible to translate this problem in a Mixed Integer Program, where the sum of the differences, i.e. $\sum_k \delta_k$, of all levels is minimized.
The disadvantage of this formulation is that a minimal number of N^2 binary variables would be needed, where N in this case is the number of customers visited by the vehicle. So the Mixed Integer Program is not usable for practical applications and heuristical

throwing policies are preferred.
An easy recognition is that, if we start with customers, which are served on every day, all of these customers will have a level difference of $\Delta_k = 0$. These customers are called *complete* and will have a perfect arrival time consistency in this throwing policy, even if we include travel times, due to the fact that all routes of the vehicle will be the same at the first stops. Also the first incomplete customer added to the vehicle will have a perfect arrival time consistency, so it is possible to ignore all complete customers in the first steps.
More general, we see that generating null levels means, that the following customers will lie on a single row and therefore they have a good arrival time consistency. The problem of maximizing the null levels can be similarly formulated in a mixed integer program as the problem of minimizing the sum of level differences and has a similar complexity of roughly N^2 boolean variables. It is also necessary to say that these two problems are not equivalent as a comparison between Figure 3.3 and 3.4, respectively Table 3.2 and 3.3 shows. In the first example, a null level is generated after adding the sixth customer but less customers lie on a single row than in the second example. Still it will be shown, that maximizing the null levels is the right way to proceed.



Cust.	Level	Differences	
7	$l_7 = (3,4,3,3,3,3)$	$\Delta_6 = 1$	$\delta_6 = 0$
6	$l_6 = (3,3,3,3,3,3)$	$\Delta_5 = 0$	$\delta_5 = 0$
5	$l_5 = (2,2,2,3,3,3)$	$\Delta_4 = 1$	$\delta_4 = 1$
4	$l_4 = (2,1,2,3,2,2)$	$\Delta_3 = 1$	$\delta_3 = 1$
3	$l_3 = (1,1,2,2,2,1)$	$\Delta_2 = 1$	$\delta_2 = 1$
2	$l_2 = (1,0,2,1,1,1)$	$\Delta_1 = 1$	$\delta_1 = 1$
1	$l_1 = (0,0,1,1,0,1)$	$\Delta_1 = 1$	$\delta_1 = 0$

Figure 3.3 & Table 3.2: Example with maximal number of null levels



Cust.	Level	Differences	
6	$l_6 = (3,4,3,3,3,3)$	$\Delta_5 = 1$	$\delta_5 = 1$
5	$l_5 = (2,3,2,3,3,3)$	$\Delta_4 = 1$	$\delta_4 = 0$
4	$l_4 = (2,2,2,3,2,2)$	$\Delta_3 = 1$	$\delta_3 = 1$
3	$l_3 = (1,2,2,2,2,1)$	$\Delta_2 = 1$	$\delta_2 = 0$
2	$l_2 = (1,1,2,1,1,1)$	$\Delta_1 = 1$	$\delta_1 = 1$
1	$l_1 = (0,1,1,1,0,1)$	$\Delta_1 = 1$	$\delta_1 = 0$
7	$l_7 = (0,1,0,0,0,0)$	$\Delta_6 = 1$	$\delta_6 = 0$

Figure 3.4 & Table 3.3: Example where customers lie on a minimal number of rows

An other simple way to generate null levels is finding matching customers, where matching means that two or more incomplete customers are equivalent to one complete customer.

Definition 3.1.2 (matching customers)**.** Customers $i_1, ..., i_k \in \mathcal{N}$ are called *matching* customers if

$$\sum_{l=1}^{k} P_{i_l,d} = 1, \ \forall d \in \mathcal{D}$$

The case gets more complicated if we want to get a null level over several rows. For this approach it is easier to work with blocks. Especially when travel times are taken back into account, the block concept, compared to the minimization of the row differences, gives more flexibility for an optimal routing, while not losing the focus on the position of a customer.

3.1.2 Blocks

A block is basically defined as an array of length D of stop lists, but at first it is easier to imagine a block as a set of customers. A block may also consist of a single customer. Additionally, we also talk about complete blocks and the height of a block.

Definition 3.1.3 (blocks)**.** The set of all possible blocks is defined as $\mathcal{B} := \mathcal{P}(\mathcal{N})\backslash\{\emptyset\}$. A block $b \in \mathcal{B}$ is called *complete*, if adding all customers of this block to empty glass columns generates a null level. The *height* H_b of a block b is defined as the maximum number of balls in a column after adding all customers of this block to empty glass columns, but it is more comprehensible using a definition getting along without the unprecisely defined terms regarding the glass columns.

$$H_{b,d} := \sum_{i \in b} P_{i,d}$$
$$H_b := \max_{d \in \mathcal{D}} H_{b,d}$$

A block $b \in \mathcal{B}$ is defined as *complete* if daily height $H_{b,d}$ remains constant, i.e. $H_{b,d_1} = H_{b,d_2}, \forall d_1, d_2 \in \mathcal{D}$. The set of complete Blocks is called $\mathcal{B}^C$.

Every complete customer and also matching customers can be treated as a complete block with height 1.

Lemma 3.1.4. *Let $i \in \mathcal{N}$ be a customer requiring service on every day, i.e. $P_{i,d} = 1, \ \forall d \in \mathcal{D}$, than $b_1 := \{i\}$ is a complete block, due to*

$$H_{b_1,d} = P_{i,d} = 1, \forall d \in \mathcal{D} \ \Rightarrow \ b_1 \in \mathcal{B}^C.$$

Subsequently, let $i_1, ..., i_k \in \mathcal{N}$ be matching customers, than $b_2 := \{i_1, ..., i_k\}$ also defines a complete block.

$$H_{b_2,d} = \sum_{l=1}^{k} P_{i_l,d} = 1, \ \forall d \in \mathcal{D} \ \Rightarrow \ b_2 \in \mathcal{B}^C.$$

Conversely, it can be seen, that generating null levels in the glass columns, creates a partition of the customers into blocks.

Lemma 3.1.5. *Let $(i_k)_{k=1,...,m}$ be a sequence of customers, $1 \leq k_1 < k_2 \leq m$ and $b := \{i_{k_1+1}, ..., i_{k_2}\} \in \mathcal{B}$, then the following equation holds.*

$$H_{b,d} = \sum_{k_0=k_1+1}^{k_2} P_{i_{k_0},d} = \sum_{k_0=1}^{k_2} P_{i_{k_0},d} - \sum_{k_0=1}^{k_1} P_{i_{k_0},d} = \lambda_{k_2,d} - \lambda_{k_1,d}.$$

Assuming $\Delta_{k_1} = \Delta_{k_2} = 0$ per definition means that $\lambda_{k_1,d_1} = \lambda_{k_1,d_2}$ and $\lambda_{k_2,d_1} = \lambda_{k_2,d_2}$ for all $d \in \mathcal{D}$ and therefore leads to $b \in \mathcal{B}^C$.
In conclusion, let $(k_l)_{l=0,...,r}$ be the strictly monotonically increasing sequence of indexes with $k_0 = 1$ and $\Delta_{k_l} = 0$ for all $l \in \{1, ..., r\}$, then the blocks defined by

$$b_l := \{i_{k_{l-1}}, ..., i_{k_l}\}, \ \forall l \in \{1, ..., r\}$$

are complete.
Conversely, assume $\bigcup_{l=1}^{r+1} b_l = \{i_1, ..., i_m\}$ as a partition of disjoint blocks and $b_l \in \mathcal{B}^C$ for all $l \in \{1, ..., r\}$. Then the sequence

$$(i_{1,1}, ..., i_{1,m_{b_1}}, i_{2,1}, ..., i_{2,m_{b_2}}, ..., i_{r+1,1}, ..., i_{r+1,m_{b_{r+1}}}),$$

with $m_{b_l} := |b_l|$ and $i_{l,k} \in b_l, \ \forall k \in \{1, ..., m_{b_l}\}$, holds that $\Delta_{l,m_{b_l}} = 0$ for all $l \in \{1, ..., r\}$.

Thus it appears that the problem of maximizing the null level is equivalent to maximizing the amount of complete blocks, but the advantage is that the term block is not necessarily bounded to glass columns any more. Therefore, it is possible assigning a block to a vehicle afterwards the blocks have been generated, giving additional optimization options (see Section 5.1).
The goal of using the concept of blocks is that after finishing the computation of blocks of customers, just the order in which the blocks should be visited by the vehicles has to be calculated and the D dimensional Consistent Vehicle Routing Problem is reduced to an one dimensional VRP. For these conventional type of Capacitated Vehicle Routing Problem several solution heuristics exist [25]. Considering also the location of the customers in our block generating algorithm does not change the amount and types of blocks, but significantly improves the arrival time consistency and reduces the total travel time of the drivers. How the blocks are generated is described in the next section.

3.1.3 Generation of complete blocks

Generating complete blocks is performed in three steps

- creating a list of possible complete blocks
- computing the best choice of blocks to achieve a partition of the customer set with a maximal number of complete blocks

- assigning the customers to the blocks regarding their location, so that all customer in one block lie as close as possible to each other

At the first step of the computation of complete blocks, it is only necessary to know on which day a customer is served. So we divide the customers by their types. During the whole generation process, complete customers are ignored, because of the reasons mentioned in Lemma 3.1.4 that these customers stand for itself as complete blocks.

Lemma 3.1.6. *The relation* $\simeq$ *defined by*

$$i \simeq j \;:\Leftrightarrow\; P_{i,d} = P_{j,d}, \forall d \in \mathcal{D}$$

is an equivalence relation.

Definition 3.1.7 (customer type)**.** With the notation of Lemma 3.1.6 two customers i and j are said to be from the same type if $i \simeq j$. Assuming a time horizon of D days results in $2^D - 1$ possible different customer types. The set of customer types is subsequently defined as the resulting quotient set $\mathcal{N}/_{\simeq}$. Every element $[i]_{\simeq} \in \mathcal{N}/_{\simeq}$ is called *customer type*.

In practise, the number of different customer types appearing in the instance is significantly smaller than the number of all customers, so it is also more productive considering problem formulations from the customer type point of view. Reducing customers to their types, directly indicates the reduction of blocks to their block types.

Definition 3.1.8 (block type)**.** Two blocks b_1 and b_2 are from the same type, if there exists a bijection π between the customers of block b_1 and the customers of b_2 in the way that i and $\pi(i)$ are from the same customer type for all customers i in b_1, i.e.

$$b_1 \simeq b_2 :\Leftrightarrow \exists \pi \in \mathfrak{Bij}(b_1, b_2) :\; i \simeq \pi(i)\ \forall i \in b_1.$$

Again it is possible defining the set of block types as $\mathcal{B}/_{\simeq}$ and a *block type* as $[b]_{\simeq}$.

Lemma 3.1.9. *If two blocks* $b_1.b_2 \in \mathcal{B}$ *are from the same type, i.e.* $b_1 \in [b_2]_{\simeq}$*, than the following conditions hold.*

$$\begin{aligned} |b_1| &= |b_2| \\ H_{b_1,d} &= H_{b_2,d},\ \forall d \in \mathcal{D} \\ H_{b_1} &= H_{b_2} \\ b_1 \in \mathcal{B}^C &\Leftrightarrow b_2 \in \mathcal{B}^C \end{aligned}$$

The second and third conditions guarantee that the height definitions hold for block types, as well. The height of a block type $[b]_{\simeq}$ *is simply defined by the height of one of its representative. The fourth condition gives the possibility of defining the set of complete block types as* $\mathcal{B}^C/_{\simeq}$.

Basically, a block type can be imagined as a multiset of customer types, because the customers in a block do not necessarily have different types. So it is not difficult to imagine what is meant by sub block type.

Definition 3.1.10 (sub block type)**.** A block type b_1 is a *sub block type* of a block type b_2, if the multiset of customer types of block b_1 is a subset of the multiset of customer types of block b_2 or more detailed

$$[b_1]_{\simeq} \leq [b_2]_{\simeq} :\Leftrightarrow \exists \pi \in \mathfrak{Inj}(b_1, b_2) : \ i \simeq \pi(i), \ \forall i \in b_1.$$

Theorem 3.1.11. *$\leq$ defines a partial order on $\mathcal{B}/_{\simeq}$ and the greatest element of the poset $(\mathcal{B}/_{\simeq}, \leq)$ is $[\mathcal{N}]_{\simeq}$.*

Proof. Assuming $[b_1]_{\simeq} \leq [b_2]_{\simeq}$ and $[b_2]_{\simeq} \leq [b_1]_{\simeq}$ leads per definition to the existence of injections $\pi_1 \in \mathfrak{Inj}(b_1, b_2)$ and $\pi_2 \in \mathfrak{Inj}(b_2, b_1)$. From the finiteness of b_1 and b_2 follows that $|b_1| = |b_2|$ and $\pi_1 \in \mathfrak{Bij}(b_1, b_2)$. This results per definition in $b_1 \simeq b_2$ and $[b_1]_{\simeq} = [b_2]_{\simeq}$ gives the antisymmetry of $\leq$.
With $\pi_1 \in \mathfrak{Inj}(b_1, b_2)$ and $\pi_2 \in \mathfrak{Inj}(b_2, b_3)$ also $\pi_3 := \pi_2 \circ \pi_1$ is an injection from b_1 to b_3 fulfilling

$$\pi_3(i) = \pi_2(\pi_1(i)) \simeq \pi_1(i) \simeq i$$

resulting in the transitivity of $\leq$.
Choosing $\pi = id$ leads on the one hand to the reflexivity of $\leq$. On the other also the maximality of $[\mathcal{N}]_{\simeq}$ in the poset $(\mathcal{B}/_{\simeq}, \leq)$ can be seen. □

Theorem 3.1.12. *For every two blocks $b_1, b_2 \in \mathcal{B}$ holds*

$$b_1 \subseteq b_2 \Rightarrow [b_1]_{\simeq} \leq [b_2]_{\simeq}$$
$$[b_1]_{\simeq} \leq [b_2]_{\simeq} \Rightarrow \exists b^* \in \mathcal{B} : b^* \subseteq b_2 \wedge b^* \in [b_1]_{\simeq}$$

Proof. The first condition results from the reflexivity of $\simeq$ and the choice of $\pi = id|_{b_1}$. Per definition $[b_1]_{\simeq} \leq [b_2]_{\simeq}$ leads to the existence of an injection $\pi \in \mathfrak{Inj}(b_1, b_2)$ fulfilling $i \simeq \pi(i)$ for all $i \in b_1$. The definition of $\mathcal{B} := \mathcal{P}(\mathcal{N})\backslash\{\emptyset\}$ as the power set of $\mathcal{N}$ gives that the set defined by $b^* = \pi(b_1) \subseteq b_2$ is a block, i.e. $b^* \in \mathcal{B}$. Additionally, it holds $\pi \in \mathfrak{Bij}(b_1, b^*)$ and therefore $b_1 \simeq b^*$. □

Although no unique least element in the poset $(\mathcal{B}/_{\simeq}, \leq)$ can be found, it is still possible asking for minimal elements. For the set $\mathcal{B}/_{\simeq}$ the minimal elements are simply the equivalence classes induced by the blocks consisting of a single customer, i.e. $[\{i\}]_{\simeq}$ for all customers i. This question gets more interesting if the considered set is $\mathcal{B}^C/_{\simeq}$.

Definition 3.1.13 (indivisible)**.** The minimal elements of the poset $(\mathcal{B}^C/_{\simeq}, \leq)$ are called *indivisible*. A complete block type $[b_1]_{\simeq}$ is called *indivisible* if $[b_2]_{\simeq} \not\leq [b_1]_{\simeq}$ for every other block type $[b_2]_{\simeq} \neq [b_1]_{\simeq}$ or equivalent

$$\forall [b_2]_{\simeq} \in \mathcal{B}^C/_{\simeq} \ : \ [b_2]_{\simeq} \leq [b_1]_{\simeq} \Rightarrow [b_2]_{\simeq} = [b_1]_{\simeq}$$

Additionally, a complete block $b_1 \in \mathcal{B}^C$ is called *indivisible* if there does not exist any complete $b_2 \in \mathcal{B}^C$ with $b_2 \subsetneq b_1$. The set of indivisible blocks is defined by $\mathcal{B}^U \subseteq \mathcal{B}^C$ and the set of indivisible block types as $\mathcal{B}^U/_\simeq$.

The last set of the definition above is well-defined because of the following conclusion.

Corollary 3.1.14. *A complete block $b \in \mathcal{B}^C$ is indivisible if and only if the induced block type $[b]_\simeq \in \mathcal{B}^C/_\simeq$ is indivisible. Therefore, the set of block types induced by indivisible blocks $\mathcal{B}^U/_\simeq$ is equal to the set of indivisible block types.*

Proof. See Definition 3.1.13 and Theorem 3.1.12. □

Considering the poset of complete block types, also discussed in Appendix A, only the minimal elements are from interest. The goal of generating as much blocks as possible with no sub blocks results in a higher flexibility in the final optimization. The existence of a divisible complete block in a partition of the customer set also implies that the choice of complete blocks is not optimal.
In the following, the generation of the set $\mathcal{B}^U/_\simeq$, of complete, indivisible block types is introduced and how the computation is implemented in an algorithm.

Generating a list of different indivisible, complete block types

The generation of complete, indivisible block types works in three phases.

- In a preprocessing step, all pairs of matching customer types are determined, where matching means that these customers complete each other in a way that they form a block type of height 1 (see Definition 3.1.2).
- All multisets of customer types representing complete block types $[b]_\simeq \in \mathcal{B}^C/_\simeq$ of a fixed height $H_b = H$ are computed. Here the maximum numbers of all block types must be considered.
- Each block type is checked on its indivisibility.

The last two steps are performed for all heights of block types H_b lower equal than a predefined maximal height of a block.

Preprocessing step The pairs of matching customer type can easily be found by using the binary type introduced and discussed in Appendix A. Considering a set of just two matching customers means that these customers are complement to each other (see Definition 3.1.2 and A.0.4). So a single equation indicates whether two customer types match or not (see Theorem A.0.2). With every pair of matching customer types a block type is generated. These block types are clearly complete and indivisible, due to the fact that all block types of height 1 are indivisible. Later it is shown that the use of the complement block type shortens the running time of the algorithm. So it is important to mention that all block types resulting from just two matching customers

are self complementary (see also Definition A.0.4). During this process the information of the complement customer type, so far it exist in the instance, is assigned to each customer type for further usage in the implementation.
Additionally, all customer types are categorized in D pairwise disjunct sets, where a customer type conditional also will be in set $\mathcal{N}_d$ with $d \in \{1, .., D\}$, if it is not served on the first $d-1$ days, i.e.

$$\mathcal{N}_d := \{[i]_{\simeq} \in \mathcal{N}/_{\simeq} \mid \forall d' < d : P_{i,d'} = 0 \ \wedge \ P_{i,d} = 1\}$$

This categorization is important in the next step.

Computation of complete Blocks Using the notation of the last paragraph the following simplification of the complete block generating algorithm explains, how a single complete block is generated.

Data: H_b , $\{\mathcal{N}_1, ..., \mathcal{N}_D\}$
Result: b
$d \leftarrow 1$;
$\Delta \leftarrow H_b$;
$(\lambda_1, ..., \lambda_D) \leftarrow (0, ..., 0)$;
while $d \leq D$ **do**
  if $\lambda_d < H_b$ **then**
    $\Delta \leftarrow H_b - \lambda_d$;
    $\mathcal{N}' \leftarrow$ *chooseSubMultiset*$(\mathcal{N}_d, \Delta)$;
    //updating and checking the block level;
    $\lambda_d \leftarrow H_b$;
    for $i \in \mathcal{N}'$ **do**
      for $d' = d + 1$ **to** D **do**
        if $P_{i,d} = 1$ **then**
          $\lambda_{d'} \leftarrow \lambda_{d'} + 1$;
          if $\lambda_{d'} > H_b$ **then**
            return NULL;
          end
        end
      end
    end
    $b \leftarrow b \cup \mathcal{N}'$;
  end
  $d \leftarrow d + 1$;
end
return b;

Algorithm 1: Generation of a complete block

The function *chooseSubMultiset*$(\mathcal{N}_d, \Delta)$ returns a multiset $\mathcal{N}'$ of customer types in $\mathcal{N}_d$ with $|\mathcal{N}'| = \Delta$ such that the multiplicity of an element $[i]_{\simeq} \in \mathcal{N}'$ is not greater than

$|[i]_{\simeq}|$. This means that also the number of customers of a specific type are considered by this algorithm.
The main difference between this simplification and the implemented algorithm lies in the choice of $\mathcal{N}'$. In practice, all choices of $\mathcal{N}'$ with $|\mathcal{N}'| = \Delta$ are taken into account. Here it is also important to prevent computing blocks from the same type, so instead of considering customers and blocks the implementation works with customer types and block types. This way, all possibilities of complete block types with height H_b are generated simultaneously by a recursive procedure. Figure 3.5 should give an image of how this recursive function works for a time horizon of three days, a height of $H_b = 2$ and five different customer types.

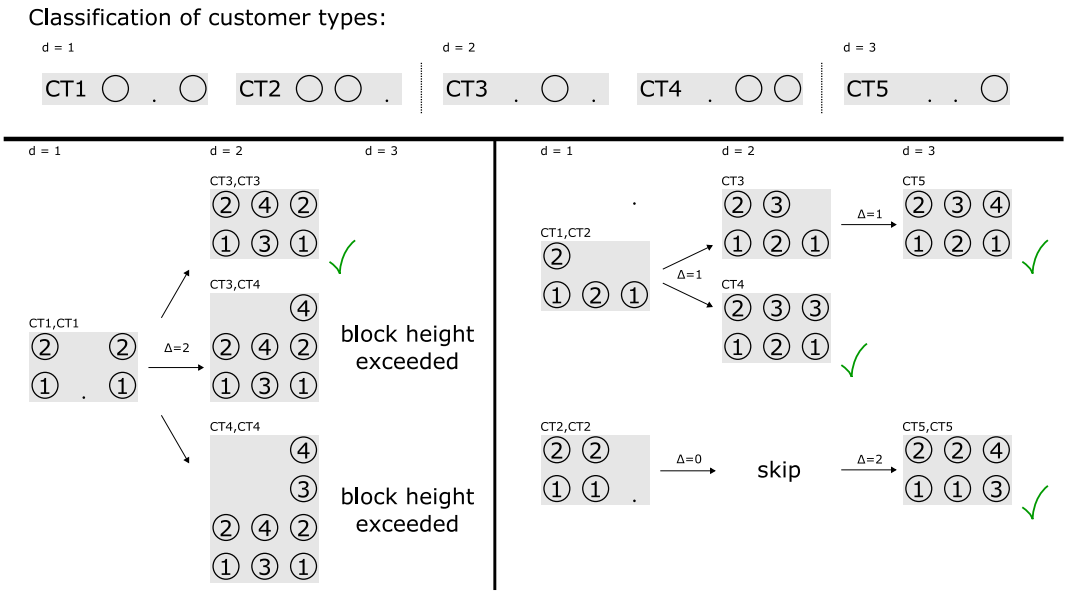


Figure 3.5: example to illustrate the block generating algorithm

The result of the implemented algorithm is a list of all complete block types with a given height H_b, which can be created with the given customer types of the instance. These block types are in many cases divisible, but it is possible to significantly shorten the list of possible block types by introducing forbidden customer types during the creation of complete blocks. Explaining this based on the simplified algorithm above this would mean that only subsets $\mathcal{N}'$ of customers are allowed, consisting of customers being not from a forbidden customers type. A customer would be forbidden if it forms a sub block with other customers already added to b. Determining this would demand the knowledge of all smaller indivisible complete block types, which in practise is very slow regarding the runtime of the algorithm.
Instead of considering all possible sub blocks, it is much faster defining the list of forbidden customer types as the list of complement customer types added to a block type

in a previous step. If two complementary customer types are part of a block type, it is not difficult to see that this block type must have a sub block type of height 1 consisting of these two customer types. Figure 3.6 illustrates the difference in the complete block generation, when forbidden customer types are added. Customers of type 1 and 3 or analogously from type 2 and 5 are a priori not allowed to be part of the same block. So instead of generating four different complete block types, which would have to be checked to be indivisible, the only complete block type remaining is, in this case, the one consisting of the customer types CT1, CT2 and CT4. This complete block type is indivisible and therefore the only indivisible block type of height $H_b = 2$, which can be built in this instance.

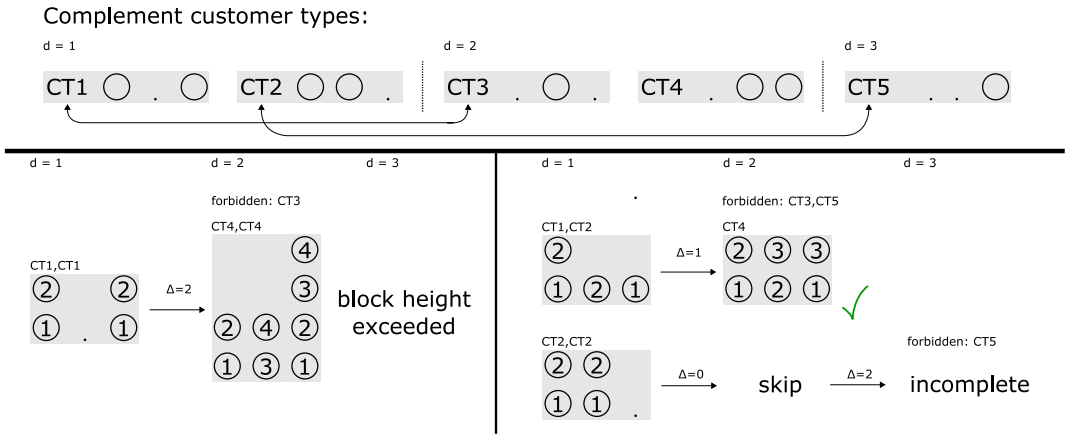


Figure 3.6: example to illustrate the improved block generating algorithm with forbidding complement customer types

More general, this way guarantees that the list of complete block types is shortened about the amount of block types, having a sub block consisting of two matching customers, meaning that significantly less block types must be controlled to be indivisible in the next step.

Testing indivisibility of complete block types Inductively, the complete block types created in the previous step are checked to have a smaller complete sub block type. Considering a complete block type $[b]_{\simeq}$ of height H_b, it is possible assuming that all indivisible block types with height smaller than H_b were computed in the previous step. The next lemma shows that a block type $[b]_{\simeq}$ of height H_b having a sub block type $[b^*]_{\simeq}$ implies $H_b \geq H_{b^*}$.

Lemma 3.1.15. *For two complete blocks $b_1, b_2 \in \mathcal{B}^C$ it holds*

$$[b_1]_{\simeq} \leq [b_2]_{\simeq} \;\Rightarrow\; H_{b_1} \leq H_{b_2}$$

Proof. Let $\pi \in \mathfrak{Inj}(b_1, b_2)$ be the injection with $i \simeq \pi(i)$ for all $i \in b_1$ and $d \in \mathcal{D}$ arbitrary then $P_{\pi(i),d} = P_{i,d}$ gives that

$$H_{b_2} = \sum_{j \in b_2} P_{j,d} = \sum_{\pi(i) \in b_2} P_{\pi(i),d} + \sum_{j \in b_2 \setminus \pi(b_1)} P_{j,d} \geq \sum_{\pi(i) \in b_2} P_{i,d} = \sum_{i \in b_1} P_{i,d} = H_{b_1}$$

□

The following theorem shows that even $H_b > H_{b^*}$ is valid and establishes a basis for Algorithm 2.

Theorem 3.1.16. *For complete blocks $b_1, b_2 \in \mathcal{B}^C$ the following implications are valid*

$$b_1 \subseteq b_2 \ \wedge \ H_{b_1} = H_{b_2} \Rightarrow b_1 = b_2$$
$$[b_1]_\simeq \leq [b_2]_\simeq \ \wedge \ H_{b_1} = H_{b_2} \Rightarrow [b_1]_\simeq = [b_2]_\simeq$$

Proof. By definition, $H_b = H_{b,d}, \ \forall d \in \mathcal{D}$ for every complete block $b \in \mathcal{B}^C$ and therefore

$$H_{b_2} = H_{b_2,d} = \sum_{i \in b_2} P_{i,d} = \sum_{i \in b_2 \setminus b_1} P_{i,d} + \sum_{i \in b_1} P_{i,d} = \sum_{i \in b_2 \setminus b_1} P_{i,d} + H_{b_1,d} = \sum_{i \in b_2 \setminus b_1} P_{i,d} + H_{b_1}$$
$$\Rightarrow \sum_{i \in b_2 \setminus b_1} P_{i,d} = H_{b_2} - H_{b_1} = 0, \ \forall d \in \mathcal{D}$$

Every customer in $\mathcal{N}$ is served at least on one day, so it follows that $b_2 \setminus b_1 = \emptyset$.
For the proof of the second condition, Theorem 3.1.12 gives that there exists a $b^* \in \mathcal{B}$ with $b^* \subseteq b_2$ and $b^* \in [b_1]_\simeq$. With b_1 also $b^* \in \mathcal{B}^C$ is complete and $H_{b^*} = H_{b_1} = H_{b_2}$. The first condition subsequently proofs that $b^* = b_2$ and therefore $[b^*]_\simeq = [b_1]_\simeq = [b_2]_\simeq$. □

Corollary 3.1.17. *Every complete block $b \in \mathcal{B}^C$ (complete block type $[b]_\simeq \in \mathcal{B}^C/_\simeq$) with height H_b is either indivisible or has a sub block (sub block type) of height strictly lower than H_b.*

Determining if an indivisible block type $[b^*]_\simeq \in \mathcal{B}^U/_\simeq$ is sub block of the given block type $[b]_\simeq \in \mathcal{B}^C/_\simeq$ is realized by checking if

$$|[b^*]_\simeq \cap [i]_\simeq| \leq |[b]_\simeq \cap [i]_\simeq|$$

is fulfilled for every $i \in b^*$. It is possible to perform this request in linear runtime because the multi set of customer types representing a block type are implemented as an ordered list.

Data: $\mathcal{B}^C/_{\simeq}$
Result: $\mathcal{B}^U/_{\simeq}$
for $[b]_{\simeq} \in \mathcal{B}^C/_{\simeq}$ **do**
  for $[b^*]_{\simeq} \in \mathcal{B}^U/_{\simeq}$ **do**
    if $[b^*]_{\simeq} \leq [b]_{\simeq}$ **then**
      break;
    end
  end
  if *not breaked* **then**
    $\mathcal{B}^U/_{\simeq} \leftarrow \mathcal{B}^U/_{\simeq} \cup \{[b]_{\simeq}\}$;
  end
end

Algorithm 2: Generating set of indivisible blocks

There are two ways to accelerate this testing process. The previous step showed the possibility of generating a less amount of block types by forbidding complementary customers in one block. This means that all indivisible block types of height 1 consisting of two customers could be skipped in the set of indivisible block types, because complete block types with such sub blocks are a priori excluded. It is also possible to shorten the second for-loop of the algorithm above by checking only the indivisible block types with a height lower equal $\frac{H_b}{2}$. This result from the fact proofed in Corollary A.0.17 that every divisible block type must have a sub block of height lower or equal $\frac{H_b}{2}$.
The other way to a faster runtime is using the result of Corollary A.0.11. If the algorithm finds a complete and indivisible block type $[b]_{\simeq}$ with height H_b consisting of $|b|$ customer types, then it is possible to add also the complement block to the list of indivisible block types. Here it is important to guarantee that a block type is not twice added to the list. This is fulfilled by using the formula of Corollary A.0.14, $H_{b^c} + H_b = |b|$. The following distinctions are added to the algorithm.

- If $H_b < \frac{|b|}{2}$ then also the complement block type is added to the list.
- If $H_b > \frac{|b|}{2}$ then the block type is simply skipped and has not to be controlled on indivisibility.
- For block types with $H_b = \frac{|b|}{2}$ the algorithm works in the same way as above.

The first condition shows that after a block type is checked to be indivisible, only bigger - in the sense of height - complement block types are added to the list of indivisible block types. This guarantees that the complete block type has not been added to the list in a previous step.
The second condition guarantees that every block type fulfilling the inequality $H_b > \frac{|b|}{2}$ can be skipped because in the case it is indivisible, it would have been added as the complement block type of a block type of height $|b| - H_b$ which is strictly lower than H_b. If a block type has a height of $\frac{|b|}{2}$ this means that the complement block type also has a height of $\frac{|b|}{2}$. In order to prevent that complications occur, these block type are treated

in the usual way.
The last important thing is keeping the list of indivisible block types ordered regarding the block height. This is done by recording the number of indivisible block types of each height and adding a block type on the right position to the list.

Calculating the choice of block types

After successfully generating a set of feasible, indivisible and complete block types $\mathcal{B}^U/_{\simeq}$, it is time to decide how the actual customer grouping in blocks should be computed and of which block types it will consist of. Such a grouping $\mathcal{B}' \subseteq \mathcal{B}$ has to fulfil $\bigcup_{b\in\mathcal{B}'} b = \mathcal{N}$ with $b_1 \cap b_2 = \emptyset$ for all $b_1, b_2 \in \mathcal{B}'$ with $b_1 \neq b_2$ or equivalent, $\mathcal{B}'$ must be a partition of $\mathcal{N}$. Although it is not possible assuming all blocks to be complete, the goal is to maximize the number of complete blocks in the grouping, i.e. $\mathcal{B}' \cap \mathcal{B}^C$, which is equivalent to maximizing the size of $\mathcal{B}' \cap \mathcal{B}^U$.

Lemma 3.1.18. *Let $\mathcal{B}' \subseteq \mathcal{B}$ be a set of blocks fulfilling $\bigcup_{b\in\mathcal{B}'} b = \mathcal{N}$ with $b_1 \cap b_2 = \emptyset$ for all $b_1, b_2 \in \mathcal{B}'$ with $b_1 \neq b_2$, such that the number of complete blocks in $\mathcal{B}'$ is maximal, then $\mathcal{B}' \cap \mathcal{B}^C \subseteq \mathcal{B}^U$.*

Proof. Assuming $b \in (\mathcal{B}' \cap \mathcal{B}^C)\backslash\mathcal{B}^U$ Theorem A.0.15 gives that b can be written as a partition of more than one indivisible blocks. Replacing b in $\mathcal{B}'$ by its partition is a contradiction to the maximality condition of $\mathcal{B}'$. □

Additionally, it is also important that the customers within a block lie geographically close to each other, which is the second part of the trade off needed to be addressed by the customer grouping. A choice of blocks with a maximal number of complete, indivisible blocks is not necessarily the best way to group the customers regarding their locality. The suggested policy is maximizing the amount of complete indivisible blocks, which guarantees flexibility for further optimization of the computed solution. A high number of blocks indirectly leads to smaller blocks regarding their height and their number of customers they are consisting of. Finally, it is also easier matching closely located customers to a block, if the block is smaller, which is treated in the next part of this section. For the sake of completeness, it is also important to say that the previously mentioned trade off can be addressed from the opposite direction, in particular choosing a $\mathcal{B}' \subseteq \mathcal{B}$ such that the geographical locations of the customers within a block lie as close as possible. The exact formulation and the solving strategy of this problem is documented in Appendix B.
Before introducing the MIP maximizing the number of blocks, it is important to summarize the information created so far. The list of indivisible, complete block types shows what kind of blocks could be generated with the customers in the regarded instance. Additionally, the number of customers in a block being from a specific customer type are a priori given, which is also a required information to form blocks.

Definition 3.1.19. Let $b \in \mathcal{B}$ and $i \in \mathcal{N}$, then the number of customers of type $[i]_\simeq$ in a block of type $[b]_\simeq$ is defined by

$$N_{b,i} := |b \cap [i]_\simeq|$$

The independence of $N_{b,i}$ from the choice of the customer in $[i]_\simeq$ is per definition fulfilled. Theorem 3.1.20 shows that $N_{b,i}$ is also independent from the choice of the representatives $b \in \mathcal{B}^U$ and therefore guarantees the well-definedness.

Theorem 3.1.20. *Given a block type $[b]_\simeq \in \mathcal{B}^U/_\simeq$ and a block $b^* \in [b]_\simeq$, then the following condition holds for every customer type $[i_0]_\simeq \in \mathcal{N}/_\simeq$.*

$$N_{b,i_0} := |b \cap [i_0]_\simeq| = |b^* \cap [i_0]_\simeq| =: N_{b^*,i_0}$$

Proof. $b \simeq b^*$ means by definition the existence of a bijective function $\pi \in \mathfrak{Bij}(b, b^*)$ fulfilling $\pi(i) \simeq i$ for every $i \in b$. It can easily be seen that also the inverse function guarantees $j \simeq \pi^{-1}(j)$ for every customer $j \in b^*$. If we additionally assume $j \in [i_0]_\simeq \cap b^*$, it follows that also $\pi^{-1}(j) \in [i_0]_\simeq$ and therefore $\pi^{-1}(j) \in b \cap [i_0]_\simeq$ and the surjectivity of $\pi|_{b\cap[i_0]_\simeq}$. The injectivity is given by the injectivity of π. In conclusion, $\pi|_{b\cap[i_0]_\simeq}$ is bijective and the sets $b \cap [i_0]_\simeq$ and $b^* \cap [i_0]_\simeq$ are from the same cardinality. □

Let $\mathcal{B}_0 \subseteq \mathcal{B}^U$ be the minimal subset of representatives such that $\mathcal{B}^U/_\simeq = \{[b]_\simeq : b \in \mathcal{B}_0\}$. The variable n_b for $b \in \mathcal{B}_0$ describes how many blocks of block type $[b]_\simeq \in \mathcal{B}^U/_\simeq$ are used for the block grouping of the initial solution.

Mixed Integer Program 5. *maximizing*

$$\sum_{b\in\mathcal{B}_0} n_b + \epsilon \sum_{\substack{[i]_\simeq\in\mathcal{N}/_\simeq \\ |[i]_\simeq|\geq 2}} \sum_{b\in\mathcal{B}_0} N_{b,i} n_b$$

subject to

$$\sum_{b\in\mathcal{B}_0} N_{b,i} n_b \leq |[i]_\simeq|, \quad \forall [i]_\simeq \in \mathcal{N}/_\simeq \tag{3.1}$$

$$n_b \in \mathbb{N} \tag{3.2}$$

If two solutions have the same amount of complete, indivisible blocks, it is the purpose of the objective function choosing the solution, where more frequently visited customers are covered by the blocks. Guaranteeing this lexicographic order the parameter ϵ is chosen to be sufficiently small. The number of blocks consisting of customers of type $[i]_\simeq$ is limited by the total amount of customers of this type, i.e. $|[i]_\simeq|$, in Constraint (3.1).
The MIP does not calculate the actual choice of complete, indivisible blocks in $\mathcal{B}' \cap \mathcal{B}^C$, where $\mathcal{B}'$ is the partition of $\mathcal{N}$ introduced at the beginning of this subsection. The only information obtained by solving MIP 5 is the multiplicity n_b of how many blocks of type $[b]_\simeq$ appear in $\mathcal{B}'$. Which customers are actually grouped together is not specified. This is the part of the next section, where customers are matched to their blocks.

Matching customers to their blocks

At the moment, it is known, how many blocks of each type we need for the customer grouping and of which customer types they are consisting of. The information getting lost by working with customer types is the location, the delivery amount and the time it takes to serve the customer, called service time. If we imagine a single customer of a specific type in a block, we now want to know what customer of our instance is actually meant by this. Every customer can only be matched to a block if its customer type is required to form this block. The next definitions should formalize the procedure of matching customers to blocks.

Definition 3.1.21. For every $b \in \mathcal{B}_0$ with $n_b > 0$, where $\mathcal{B}_0$ is the minimal set of representatives of $\mathcal{B}^U/_{\simeq}$ introduced in the previous subsection and n_b are the multiplicities calculated in MIP 5, the artificial blocks

$$\mathbf{b}_k^{(b)},\ k \in 1, ..., n_b$$

are called *empty blocks*. The set of all empty blocks is described by

$$\mathbf{B}^E := \{\mathbf{b}_k^{(b)}\ :\ k \in 1, ..., n_b\ \wedge\ b \in \mathcal{B}_0\ \wedge\ n_b > 0\}.$$

The equivalence relation $\simeq$ is enhanced to $\mathcal{B} \dot{\cup} \mathbf{B}^E$ by defining

$$\mathbf{b}_k^{(b)} \simeq b^* \Leftrightarrow b \simeq b^*$$
$$\mathbf{b}_{k_1}^{(b_1)} \simeq \mathbf{b}_{k_2}^{(b_2)} \Leftrightarrow b_1 \simeq b_2$$

This relationship between customers and empty blocks can be imagined in a bipartite, undirected graph, where the set of empty blocks $\mathbf{B}^E$ and the set of customers $\mathcal{N}$ are the two classes of the bipartite graph and $\mathcal{E}$ is defined as the set of edges between these two. An empty block $\mathbf{b} \in \mathbf{B}^E$ is connected to a customer i, if there exists a block b with $b \simeq \mathbf{b}$ containing customer i, i.e.

$$(\mathbf{b}, i) \in \mathcal{E} :\Leftrightarrow \exists b\ :\ b \simeq \mathbf{b}\ \wedge\ i \in b.$$

Figure 3.7 gives an overview of the introduced notation. The left side shows two empty blocks representing indivisible block types $[b_1]_{\simeq}, [b_2]_{\simeq} \in \mathcal{B}^U/_{\simeq}$ with $n_{b_1} > 0$ and $n_{b_2} > 0$ computed by the MIP 5. Both of the block types only appear with a multiplicity $n_{b_1} = n_{b_2} = 1$, because the first block type is limited by the amount of customers of type 3 and the second one by the amount of customers of type 4. Additionally, the sum of the amounts of both block types is also restricted by the number of customers of type 2. On the right side of this example we see 6 different customers of 4 different customer types, each of them connected to the empty blocks they are "fitting" in.
At the beginning of this section, the following condition holds to every empty block of the graph: a block $\mathbf{b} \in \mathbf{B}^E$ which is connected to a customer $i \in \mathcal{N}$, is also connected to every other customer j of the same type as customer i, i.e. $\forall j \in [i]_{\simeq}, \forall \mathbf{b} \in \mathbf{B}^E : (\mathbf{b}, i) \in \mathcal{E} \Rightarrow (\mathbf{b}, j) \in \mathcal{E}$ (see also Figure 3.7). This leads to the definition of a bundle of edges.

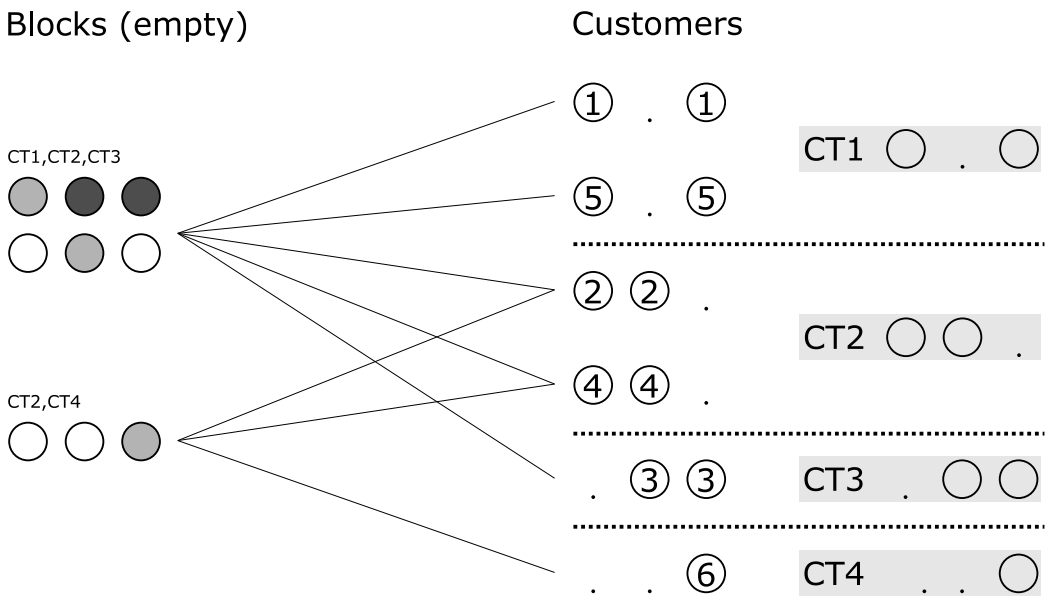


Figure 3.7: example of a bipartite graph of empty blocks and the connected customers

Definition 3.1.22. A *bundle of edges* $(\mathbf{b}, i)_{\simeq}$, respectively $(\mathbf{b}, i)_{\simeq}^{\mathcal{E}_0}$ describing a bundle of edges regarding an edge set $\mathcal{E}_0 \subseteq \mathcal{B}_0 \times \mathcal{N}$, are then defined as

$$(\mathbf{b}, i)_{\simeq} := \{(\mathbf{b}, j) : j \in [i]_{\simeq}\}$$
$$(\mathbf{b}, i)_{\simeq}^{\mathcal{E}_0} := (b, i)_{\simeq} \cap \mathcal{E}_0$$

The set of bundles of edges in $\mathcal{E}_0$ will be denoted as $\mathcal{E}_0/_{\simeq}$.

Remark 3.1.23. *As previously discussed, an empty block* $\mathbf{b}_k^{(b)}$ *may represent a block type* $[b]_{\simeq} \in \mathcal{B}^U/_{\simeq}$ *consisting of more than one customers of type* $[i]_{\simeq}$*. This amount was defined by the constant* $N_{b,i}$*. For the sake of a simplification of the notation,* $N_{b,i}$ *is also enhanced for empty blocks* $\mathbf{b} \in \mathbf{B}^E$*. Let* $\mathbf{b} \in \mathbf{B}^E$ *and* $b^* \in \mathcal{B}$ *be an arbitrary block with* $b^* \simeq \mathbf{b}$ *then*

$$N_{\mathbf{b},i} := N_{b^*,i}.$$

Analogously, the multiplicities calculated by MIP 5 are enhanced and defined for empty blocks by

$$n_{\mathbf{b}} := n_{b^*}.$$

The well-definedness is guaranteed by Theorem 3.1.20 and the definition of $\simeq$ *for empty blocks in Definition 3.1.21.*

If we now want to match the customers to their blocks, this means that in every bundle of edges $(\mathbf{b}, i)_{\simeq}$ exactly $N_{\mathbf{b},i}$ edges have to be chosen, respectively $N_{\mathbf{b},i}$ customers of type $[i]_{\simeq}$ have to be matched to the empty block $\mathbf{b}$.
As mentioned earlier, this matching should be fulfilled in the best way regarding the locations of the customers. All customers in a block should be as close as possible to each other. So we are talking about a minimization problem which can be described and solved in an MIP. The related difficulty is measuring the inner distances between the customers in a block, in the case the customers in one block are not known a priori. Using the pairwise distances between two customers would result in a quadratic amount of variables and constraints. Also the measurement parameter describing the locality of a block, consisting of a higher amount of customers, would be overpenalized in the objective function. The more efficient way is using an artificial centre of each block and measuring the distance of each customer to this centre with the Manhattan metric. This metric measures the distance between two dimensional points as it is done in a quadratic grid.
For the sake of convenience, the two coordinates of a customer i are the latitude G_i^{lat} and the longitude G_i^{lon} of its GPS position. A boolean variable called $x_{\mathbf{b},i}$, related to the edge $(\mathbf{b}, i) \in \mathcal{E}$, decodes if a customer i is matched to the empty block $\mathbf{b}$. Important to mention is that also the artificial centre of the block is not known a priori. The artificial centre of an empty block $\mathbf{b}$ is therefore introduced to the MIP by adding two auxiliary variables called $m_{\mathbf{b}}^{lat}$ and $m_{\mathbf{b}}^{lon}$. The distance between a customer and the artificial centre of a block is measured by the variables $g_{\mathbf{b},i}^{lat}$ and $g_{\mathbf{b},i}^{lon}$ and minimized in the objective function.

Mixed Integer Program 6. *minimizing*

$$\sum_{(b,i)\in\mathcal{E}} (g_{\mathbf{b},i}^{lat} + g_{\mathbf{b},i}^{lon})$$

subject to

$$g_{\mathbf{b},i}^{lat} \geq G_i^{lat} - m_{\mathbf{b}}^{lat} - M(1 - x_{\mathbf{b},i}), \qquad \forall (\mathbf{b}, i) \in \mathcal{E} \tag{3.3}$$

$$g_{\mathbf{b},i}^{lat} \geq m_{\mathbf{b}}^{lat} - G_i^{lat} - M(1 - x_{\mathbf{b},i}), \qquad \forall (\mathbf{b}, i) \in \mathcal{E} \tag{3.4}$$

$$g_{\mathbf{b},i}^{lon} \geq G_i^{lon} - m_{\mathbf{b}}^{lon} - M(1 - x_{\mathbf{b},i}), \qquad \forall (\mathbf{b}, i) \in \mathcal{E} \tag{3.5}$$

$$g_{\mathbf{b},i}^{lon} \geq m_{\mathbf{b}}^{lon} - G_i^{lon} - M(1 - x_{\mathbf{b},i}), \qquad \forall (\mathbf{b}, i) \in \mathcal{E} \tag{3.6}$$

$$1 \geq \sum_{\mathbf{b}\in\mathbf{B}:\ (\mathbf{b},i)\in\mathcal{E}} x_{\mathbf{b},i}, \qquad \forall i \in \mathcal{N} \tag{3.7}$$

$$N_{\mathbf{b},i} = \sum_{j\in[i]_{\simeq}} x_{\mathbf{b},j}, \qquad \forall (\mathbf{b}, i)_{\simeq} \in \mathcal{E}/_{\simeq} \tag{3.8}$$

$$g_{\mathbf{b},i}^{lon}, g_{\mathbf{b},i}^{lat} \geq 0, \qquad \forall (\mathbf{b}, i) \in \mathcal{E} \tag{3.9}$$

$$m_{\mathbf{b}}^{lon}, m_{\mathbf{b}}^{lat} \geq 0, \qquad \forall \mathbf{b} \in \mathbf{B}^E \tag{3.10}$$

$$x_{\mathbf{b},i} \in \{0, 1\}, \qquad \forall (\mathbf{b}, i) \in \mathcal{E} \tag{3.11}$$

The inequations (3.3) and (3.4) guarantee that $g^{lat}_{\mathbf{b},i} = |m^{lat}_{\mathbf{b}} - G^{lat}_i|$ if customer i is matched to block b and $g^{lat}_{\mathbf{b},i} = 0$ otherwise. (3.5) and (3.6) are the analogues to (3.3) and (3.4) regarding the longitude distances. That every customer is matched to at most one block is fulfilled by Inequation (3.7). Constraint (3.8) checks that every block gets the correct amount of customers of the right type. (3.9) to (3.11) are the non-negativity and binary constraints.
The artificial centre of each block, described by the variables $m^{lat}_{\mathbf{b}}$ and $m^{lon}_{\mathbf{b}}$ in the MIP, will attain the value with minimal distance to the GPS coordinates of the customers within the block $\mathbf{b}$. Measuring this distance with the Euclidean Metric would mean that the artificial centre is equal the geometric median of the customer's locations. Regarding the Manhattan Metric this minimal point can be found by calculating the 1-dimensional geometric median for every coordinate, which is equal to the conventional median as it is shown in Appendix C. In edition, the variables $m^{lat}_{\mathbf{b}}$ and $m^{lon}_{\mathbf{b}}$ will attain the following values.

$$\begin{aligned} m^{lat}_{\mathbf{b}} &= \text{median}\{G^{lat}_i : x_{\mathbf{b},i} = 1\} \\ m^{lon}_{\mathbf{b}} &= \text{median}\{G^{lon}_i : x_{\mathbf{b},i} = 1\} \end{aligned}$$

This linear mixed integer program consists of $2|\mathcal{E}|+|\mathbf{B}^E|$ continuous variables, $|\mathcal{E}|$ binary variables and $4|\mathcal{E}| + N + |\mathcal{E}/_{\simeq}|$ constraints if the non-negativity and binary constraints are ignored. So the amount of edges is the key factor influencing the runtime of the solving algorithm. In practice, the amount of edges may be too large for directly using the MIP above, resulting in the necessity of reducing the set of edges. This preprocessing step is called *fixing edges*.

Fixing edges *Fixing an edge* means that a specific edge in a bundle of edges is chosen, or equivalent a customer is matched to an empty block. This happens before the optimization process and the solving of the mixed integer program. If an edge between a customer i and an empty block $\mathbf{b}$ is fixed, it is necessary to delete some other edges fulfilling the following rules.

- every customer can be matched to at most one empty block $\mathbf{b}$
- every empty block $\mathbf{b}$ must be matched to exactly $N_{\mathbf{b},i}$ customers of type $[i]_{\simeq}$

The first point means that all other edges $(\mathbf{b}', i) \in \mathcal{E}$ with $\mathbf{b}' \neq \mathbf{b}$ are deleted from the set of edges. When $N_{\mathbf{b},i}$ customers of type $[i]_{\simeq}$ have been matched to an empty block $\mathbf{b}$, than every edge $(\mathbf{b}, j) \in (\mathbf{b}, i)_{\simeq}$ which has not been fixed to block $\mathbf{b}$ before, has to be removed from the graph in the second case. Also the edge $(\mathbf{b}, i)$ itself is deleted from $\mathcal{E}$ and added to the set of fixed edges called $\mathcal{F}$. Every time an edge is fixed, the set of edges $\mathcal{E}$ is replaced by a subset, so the condition $(\mathbf{b}, i)_{\simeq} = (\mathbf{b}, i)^{\mathcal{E}}_{\simeq}$, does not hold any more after an edge of this equivalence class is fixed.

Data: $\mathcal{E}$, $\mathcal{F}$, $(\mathbf{b}, i) \in \mathcal{E}$
$\mathcal{F} \leftarrow \mathcal{F} \cup \{(\mathbf{b}, i)\}$;
for $(\mathbf{b}', j) \in \mathcal{E}$ **do**
  if $j = i$ **then**
    $\mathcal{E} \leftarrow \mathcal{E} \backslash \{(\mathbf{b}', j)\}$;
  end
end
if $|(\mathbf{b}, i)^{\mathcal{F}}_{\simeq}| = N_{\mathbf{b},i}$ **then**
  for $(\mathbf{b}', j) \in \mathcal{E}$ **do**
    if $(\mathbf{b}', j) \simeq (\mathbf{b}, i)$ **then**
      $\mathcal{E} \leftarrow \mathcal{E} \backslash \{(\mathbf{b}', j)\}$;
    end
  end
end

Algorithm 3: Edge fixing

Let $\mathcal{E}'$ and $\mathcal{F}' = \mathcal{F} \cup \{(\mathbf{b}, i)\}$ be the edge sets after fixing an edge $(\mathbf{b}, i)$, then the worst case regarding the number of edges is $|\mathcal{E}| + |\mathcal{F}| = |\mathcal{E}'| + |\mathcal{F}'| = |\mathcal{E}'| + |\mathcal{F}| + 1$ thus $|\mathcal{E}| - 1 = |\mathcal{E}'|$ and $\mathcal{E}' = \mathcal{E} \backslash \{(\mathbf{b}, i)\}$. This would mean that no other edge in $\mathcal{E}$ is removed, which is a rarity.
The following conditions give two possibilities of fixing a customer to a block without influencing the solution of MIP 6.

$$N_{\mathbf{b},i} - |(\mathbf{b}, i)^{\mathcal{F}}_{\simeq}| = |(\mathbf{b}, i)^{\mathcal{E}}_{\simeq}| \tag{3.12}$$

$$n_{\mathbf{b}} = |[i]_{\simeq}| \quad \wedge \quad \nexists (\mathbf{b}', j) \in \mathcal{F} : \mathbf{b} \simeq \mathbf{b}' \tag{3.13}$$

The left side of Equation (3.12) describes how many customers of type $[i]_{\simeq}$ must still be matched to block $\mathbf{b}$. If this number is equal to number of edges in $\mathcal{E}$ from empty block $\mathbf{b}$ to a customer of type $[i]_{\simeq}$, this means that there is no other choice of matching for this block in the MIP and all edges in $(\mathbf{b}, i)^{\mathcal{E}}_{\simeq}$ can be fixed. $n_{\mathbf{b}}$ in Condition (3.13) describes how many blocks b with $b \simeq \mathbf{b}$ exist in the partition $\mathcal{B}'$ of $\mathcal{N}$. If its value is equal to the number of customers of type $[i]_{\simeq}$ and no other customer was matched to an empty block in $\mathbf{B}^E$ of the same type, this means that every empty block has to be matched to exactly one customer of type $[i]_{\simeq}$. Because no other customer has been matched to these blocks before, it is possible to choose one specific matching of the customers in $[i]_{\simeq}$ without loss of generality and fixing $|[i]_{\simeq}|$ edges of the graph.
This methods of fixing edges always guarantee that MIP 6 keeps feasible. For all fixed edges it is now possible to set the boolean variables $x_{\mathbf{b},i}$ to one and for all deleted edges to zero, but it is better getting rid of unnecessary variables and constraints and formulate an adaption of MIP 6, like it is done in the following (see MIP 7).

Solving the adapted MIP The next step is formulating an equivalent MIP regarding the edge sets $\mathcal{E}$ and $\mathcal{F}$. For all fixed edges $(\mathbf{b}, i) \in \mathcal{F}$ the boolean variables $x_{\mathbf{b},i}$ are not needed any more. If an empty block $b \in \mathbf{B}^E$ does not appear in the edge set $\mathcal{E}$, this means

that this block is *completely fixed*. For these empty blocks the set of customers is already found and fixed in the previous step of edge fixing. So variables in the MIP regarding a completely fixed block can also be left out. For the sake of simplification of the notation, it is assumed that the set of fixed edges $\mathcal{F}$ does not contain edges connecting completely fixed blocks with their customers, i.e.

$$\forall(\mathbf{b}, i) \in \mathcal{F}\ \exists(\mathbf{b}', j) \in \mathcal{E} : \mathbf{b} = \mathbf{b}'.$$

Important to mention is that the variables $g^{lat}_{\mathbf{b},i}$ and $g^{lon}_{\mathbf{b},i}$ must been kept for fixed edges to not falsify the median centres of the blocks.

Mixed Integer Program 7. *minimizing*

$$\sum_{(\mathbf{b},i)\in\mathcal{E}\cup\mathcal{F}_0} (g^{lat}_{\mathbf{b},i} + g^{lon}_{\mathbf{b},i})$$

subject to

$$g^{lat}_{\mathbf{b},i} \geq G^{lat}_i - m^{lat}_{\mathbf{b}} - M(1 - x_{\mathbf{b},i}), \qquad \forall(\mathbf{b}, i) \in \mathcal{E} \tag{3.14}$$

$$g^{lat}_{\mathbf{b},i} \geq m^{lat}_{\mathbf{b}} - G^{lat}_i - M(1 - x_{\mathbf{b},i}), \qquad \forall(\mathbf{b}, i) \in \mathcal{E} \tag{3.15}$$

$$g^{lon}_{\mathbf{b},i} \geq G^{lon}_i - m^{lon}_{\mathbf{b}} - M(1 - x_{\mathbf{b},i}), \qquad \forall(\mathbf{b}, i) \in \mathcal{E} \tag{3.16}$$

$$g^{lon}_{\mathbf{b},i} \geq m^{lon}_{\mathbf{b}} - G^{lon}_i - M(1 - x_{\mathbf{b},i}), \qquad \forall(\mathbf{b}, i) \in \mathcal{E} \tag{3.17}$$

$$g^{lat}_{\mathbf{b},i} \geq G^{lat}_i - m^{lat}_{\mathbf{b}}, \qquad \forall(\mathbf{b}, i) \in \mathcal{F} \tag{3.18}$$

$$g^{lat}_{\mathbf{b},i} \geq m^{lat}_{\mathbf{b}} - G^{lat}_i, \qquad \forall(\mathbf{b}, i) \in \mathcal{F} \tag{3.19}$$

$$g^{lon}_{\mathbf{b},i} \geq G^{lon}_i - m^{lon}_{\mathbf{b}}, \qquad \forall(\mathbf{b}, i) \in \mathcal{F} \tag{3.20}$$

$$g^{lon}_{\mathbf{b},i} \geq m^{lon}_{\mathbf{b}} - G^{lon}_i, \qquad \forall(\mathbf{b}, i) \in \mathcal{F} \tag{3.21}$$

$$1 \geq \sum_{\mathbf{b}\in\mathbf{B}^E:\ (\mathbf{b},i)\in\mathcal{E}} x_{\mathbf{b},i}, \qquad \forall i \in \mathcal{N} \tag{3.22}$$

$$N_{\mathbf{b},i} - |(\mathbf{b}, i)^{\mathcal{F}}_{\simeq}| = \sum_{j\in[i]_\simeq:(\mathbf{b},j)\in\mathcal{E}} x_{\mathbf{b},j}, \qquad \forall(\mathbf{b}, i)_\simeq \in \mathcal{E}/_\simeq \tag{3.23}$$

$$g^{lon}_{\mathbf{b},i}, g^{lat}_{\mathbf{b},i} \geq 0, \qquad \forall(\mathbf{b}, i) \in \mathcal{E} \cup \mathcal{F} \tag{3.24}$$

$$m^{lon}_{\mathbf{b}}, m^{lat}_{\mathbf{b}} \geq 0, \qquad \forall\mathbf{b} \in \mathbf{B}^E \tag{3.25}$$

$$x_{\mathbf{b},i} \in \{0, 1\}, \qquad \forall(\mathbf{b}, i) \in \mathcal{E} \tag{3.26}$$

Constraints (3.18) to (3.21) are the equivalents of constraints (3.14) to (3.17) for fixed edges. It can be seen that these constraints get rid of the Big-M constant and only contain continuous variables. The difference of Inequation (3.22) is that the index block set of the sum slightly changed to $\{\mathbf{b} \in \mathbf{B}^E :\ (\mathbf{b}, i) \in \mathcal{E}\}$. In Constraint (3.23), it is important subtracting the number of customers of type $[i]_\simeq$ already fixed to block $\mathbf{b}$ - equal to the size of the bundle of edges regarding $\mathcal{F}$ - from $N_{\mathbf{b},i}$.
One advantage of this adaption of the MIP is the gain of more flexibility regarding the

reduction of edges. If an empty block is completely fixed, it does not appear in any edge of $\mathcal{E}$ any more, so it is possible to ignore this block. As mentioned above, the worst case in edge fixing is the exchange of a single edge from $\mathcal{E}$ to $\mathcal{F}$ while the value of $|\mathcal{E}| + |\mathcal{F}|$ remains the same. Even though this exchange means an improvement for the MIP. One boolean variable can be saved changing the sums in the constraints (3.22) and (3.23). Additionally, four Big-M constraints are replaced by four continuous constraints.
If the MIP is still too complex after fixing all possible edges without influencing the solution of MIP 6, the only way to reduce additional edges is a heuristical approach.

Heuristical edge removing and fixing While fixing edges in a heuristical way, it is important to keep the focus on the goal, that all customers in a block shall geographically lie as close to each other as possible. Measuring this locality results in the difficulty of defining the distance to an artificial geometric centre of an empty block if the customers in that block are not a priori known, i.e. especially if in a previous step no customer was fixed to this block. In this case, a heuristical assigning of first customers to all empty blocks is necessary, which is done in a way effecting the solution of MIP 6 as less as possible.

- *Step 1:* choose block $b \in \mathcal{B}_0$ in the representatives of $\mathcal{B}^U/_{\simeq}$ with maximal multiplicity $n_b > 0$ and fulfilling

$$\forall i \in \mathcal{N}, \forall k \in \{1, ..., n_b\} \ : \ (\mathbf{b}_k^{(b)}, i) \notin \mathcal{F}$$

- *Step 2:* choose customer type $[i]_{\simeq}$ with minimal $|[i]_{\simeq}|$ and

$$\exists i \in [i]_{\simeq} \ : \ (\mathbf{b}, i) \in \mathcal{E}$$

- *Step 3:* choose $n_{\mathbf{b}}$ customers of type $[i]_{\simeq}$ and arbitrarily matching them to the empty blocks $\mathbf{b}_1^{(b)}, ..., \mathbf{b}_{n_b}^{(b)}$ in $\mathbf{B}^E$

- *Step 4:* repeat until no block fulfilling the condition of *Step 1* can be found

The way of choosing the customer type with minimal $|[i]_{\simeq}|$ in *step 2* minimizes the risk of fixing edges, which would not be fixed by MIP 6. Additionally, a mean distance from a customer of type $[i]_{\simeq}$ to all other possible block members can be calculated. Then only the n_b customers with the lowest mean distance values are chosen to be assigned to the blocks in *step 3*.
Because of the fact that each block is now fixed to at least one customer, it is possible giving each edge in $\mathcal{E}$ an artificial distance. If the amount of edges is still to high to solve MIP 6 in acceptable running time after this first heuristic edge fixing step, there are still two different strategies of reducing $\mathcal{E}$.

- fixing edges with smallest distance

- removing edges with largest distance

The first way significantly reduces the size of the MIP. Therefore, it also strongly influences the solution space. The second approach has a much lower impact on the solution of MIP 6, but has the disadvantage that removing a wrong edge may result in the infeasibility of the MIP. This leads to the necessity of an algorithm deciding if an edge is removable or not.
The implementation of this thesis uses a combination of both strategies and stops, until the amount of edges do not exceed the predefined limit any more.

Finally, it is possible to assume that a solution of MIP 7 has been found in a reasonable time span and every empty block can now conversely be interpreted as an actual block.

$$b_{\mathbf{b}} := \{i \in \mathcal{N} \ : \ (\mathbf{b}, i) \in \mathcal{F}\} \dot{\cup} \{i \in \mathcal{N} \ : \ (\mathbf{b}, i) \in \mathcal{E} \wedge x_{\mathbf{b},i} = 1\}$$

So the search for the partition $\mathcal{B}'$ of $\mathcal{N}$ can be finalized by setting

$$\mathcal{B}' = \{b_{\mathbf{b}} \ : \ \mathbf{b} \in \mathbf{B}^E\} \dot{\cup} \{\{i\} \ : \ \forall \mathbf{b} \in \mathbf{B}^E i \notin b_{\mathbf{b}}\}$$

3.1.4 Formulating an initial Solution

In this part, all kinds of blocks are taken into account, i.e. all blocks consisting of a single complete customer and the previously calculated set of complete, indivisible blocks. Every customer, which could not be matched to a complete block in Section 3.1.3, forms a not necessarily complete, singleton block, consisting of just this customer. For the sake of simplicity, a block was imagined as a set of customers so far. But as mentioned at the beginning of Section 3.1.2 a block is an array of stop list of length D. So the order in which the customers of a block are served on every day is still missing. This order can be calculated separately for every day by an arbitrary travelling salesman algorithm or by solving an MIP with a complexity depending on the height of the block. Given the inner order of each block, a sequence of blocks can be directly interpreted as the routing plan of a vehicle,

$$\mathfrak{R}_v := \{(i, j, d) \in \mathcal{N}_0 \times \mathcal{N}_0 \times \mathcal{D} \mid v \text{ drives from } i \text{ to } j \text{ on day } d\}$$

Assuming a perfect driver consistency means that every customer is strictly visited by a single vehicle, resulting in the possibility that the total costs of the objective in MIP 4 can be calculated route-wise. The following LP computes the exact arrival times at the customers and the total costs of a single routing plan. For the sake of completeness, the constant parameter representing the fix costs of a vehicle is kept in the objective. Conversely, the assumption of a perfect driver consistency leads to the fact that β does not appear in the objective function any more (compare with MIP 4). Subsequently, the total costs of a solution are simply obtained by summing up the objective values of every routing plan.

Linear Program 8. *minimizing*

$$\sum_{d\in\mathcal{D}}(a_{v,d}^{end}-a_{v,d}^{start})+\alpha+\gamma\sum_{i\in\mathcal{N}}l_i \tag{3.27}$$

subject to

$$\begin{aligned}
a_{i,d}+S_{i,d}+T_{i,j} &\leq a_{j,d}, && \forall i\in\mathcal{N}, j\in\mathcal{N}, d\in\mathcal{D}:\ (i,j,d)\in\mathfrak{R}_v && (3.28)\\
a_{i,d}+S_{i,d}+T_{i,N+1} &\leq a_{v,d}^{end}, && \forall (i,N+1,d)\in\mathfrak{R}_v && (3.29)\\
a_{i,d}-T_{0,i} &\geq a_{v,d}^{start}, && \forall (0,i,d)\in\mathfrak{R}_v && (3.30)\\
a_{i,d_2}-a_{i,d_1} &\leq l_i, && \forall i\in\mathcal{N}, d_1\neq d_2 && (3.31)\\
a_{i,d},\ a_{v,d}^{start},\ a_{v,d}^{end} &\in [T_{min},T_{max}], && \forall i,d && (3.32)\\
l_i &\in [0,T_{max}-T_{min}], && \forall i && (3.33)
\end{aligned}$$

The formulation of the ConVRP in this thesis allows waiting times and different departure times of the vehicle from the depot. Therefore, it is possible optimizing the arrival times at the customers to improve the arrival time consistency. Important to mention is that measuring arrival time consistency with the method presented in [12] (see also Section 2.1.1), would result in the necessity of using an MIP for the cost calculation instead of LP 8.

Remark 3.1.24. *Fixing the departure times from the depot for every vehicle and forbidding waiting times, would guarantee that the arrival times at each stop could be calculated directly without the use of LP 8. The variables $a_{v,d}^{start}$ would then be constant and $a_{i,d}+S_{i,d}+T_{i,j}=a_{j,d}$ if the vehicle drives from customer i to j on day d. So every arrival time can be deduced from the routing plan and therefore also the costs of arrival time consistency however it is measured and the total travel time.*

With this a priori cost calculation, it is easy formulating a greedy algorithm to generate an initial solution.

```
Data: B'
V ← ∅;
for b ∈ B' do
    Δ = +∞;
    v_best ← new vehicle;
    pos_best ← 0;
    for v ∈ V ∪ {v_best} do
        for pos ∈ {0, ..., |v|} do
            if adding b to v at pos is valid then
                δ ← get cost difference if adding b to v at pos using LP 8;
                if Δ > δ then
                    Δ ← δ;
                    v_best ← v;
                    pos_best ← pos;
                end
            end
        end
    end
    add b to v_best at pos_best;
    if v_best ∉ V then
        V ← V ∪ {v_best};
    end
end
return V;
```

Algorithm 4: Generation of initial routing plans

For every block b, the best of all possible positions pos_{best} overall routes is computed, while checking, if adding block b to vehicle v at position pos is valid, is performed under the regards of all given vehicle specific limits, like load volume and maximal serving time. Additionally, there is always the possibility of adding b to a new vehicle, if it is efficient from the cost point of view or adding to an other vehicle is not possible.
An additional, crucial impact factor on the performance of this greedy algorithm is the chronology in which the blocks of $\mathcal{B}'$ are added to the vehicles. This thesis suggests starting with all complete customer blocks, followed by the computed set of indivisible blocks and treating the singleton blocks of the remaining customers at last. This way, the effect on arrival time consistency is kept as small as possible.
In summary, the computed set of routing plans can be directly translated in an initial, feasible solution of MIP 4 with perfect driver consistency.

3.2 Solving the Consistent Vehicle Routing Problem

As mentioned in Section 3.1.4, it is necessary computing an order of the stops within a block, before a further usage of blocks can happen. For the sake of comprehensibility,

the next remark should highlight, what is actually meant by the set of blocks $\mathcal{B}'$.

Remark 3.2.1. *The set of blocks $\mathcal{B}'$ considered till the end of this thesis is a partition of ordered sets of $\mathcal{N}$, i.e.*

$$\dot{\bigcup_{b \in \mathcal{B}'}} b = \mathcal{N}$$

Given this inner order results in the possibility of defining the following parameters for each block in $\mathcal{B}'$.

Definition 3.2.2. block specifics

$\zeta_{b,d}$ ... first customer in b being served on day d
$\eta_{b,d}$ ... last customer in b being served on day d
$Q_{b,d}$ ... total demand of block b on day d
$S_{b,d}$ ... total inner block service time on day d
$T_{b_1,b_2,d}$ ... travel time between two blocks on day d

The total demand $Q_{b,d}$ is basically the sum of the demands of the customers in block b being supplied on day d, i.e. $Q_{b,d} = \sum_{i \in b} P_{i,d} Q_{i,d}$. The total service time $S_{b,d}$ of a block b is the minimal time it takes to deliver all customers on day d including the total travel time it takes to visit the customers in the precomputed order. The travel time between two blocks b_1 and b_2 on day d depends on the last customer of block b_1 and the first stop of b_2, i.e.

$$T_{b_1,b_2,d} := \begin{cases} T_{\eta_{b_1,d},\zeta_{b_2,d}} & \text{, if } \eta_{b_1,d} \text{ and } \zeta_{b_2,d} \text{ exist} \\ 0 & \text{, otherwise} \end{cases}$$

Thinking about the blocks consisting of a single, not complete customer, which could not be assigned to a block in Section 3.1.3, there are days on which the customer and therefore also the block itself does not require service, meaning that neither a first nor a last stop exists on these days. This results in the fact that formulating a VRP on blocks in the usual way is only possible for blocks with a well-defined travel time, fulfilling the condition $\forall d \in \mathcal{D}\ \exists i \in b:\ P_{i,d} = 1$. The next section shows a possibility of formulating a VRP for all kinds of blocks.

3.2.1 Vehicle Routing Problems on blocks

At the beginning of this section, it is additionally assumed that $\mathcal{B}'$ fulfils the condition, that on every day there exists a customer requiring service on this specific day, i.e.

$$\forall b \in \mathcal{B}'\ \forall d \in \mathcal{D}\ \exists i \in b:\ P_{i,d} = 1$$

To formulate a VRP on blocks it is necessary adding artificial depot blocks to the set of blocks $\mathcal{B}'$, in this case the two depot blocks b_{start} and b_{end}. The travel times are

intuitively defined as $T_{b_{\text{start}},b,d} := T_{0,\zeta_{b,d}}$ and $T_{b,b_{\text{end}},d} := T_{\eta_{b,d},N+1}$. The node set of the routing graph $\mathcal{G}^B := (\mathcal{B}^+, \mathcal{A}^B := \mathcal{B}^+ \times \mathcal{B}^+)$ is defined by $\mathcal{B}^+ := \mathcal{B}' \cup \{b_{\text{start}}, b_{\text{end}}\}$. Analogously to the notation in MIP 4 and Section 2.1, variables of the Vehicle Routing Problems on blocks are defined in the following way.

Definition 3.2.3. Variables

boolean:

x_{b_1,b_2} ... representing if the arc $(b_1, b_2) \in \mathcal{A}^B$ is used

continuous:

$a_{b,d}$ ... arrival time at block b on day d
$a^{\text{end}}_{b,d}$ ... arrival time at the arrival depot on day d after visiting b
$a^{\text{start}}_{b,d}$ ... departure time at the departure depot on day d before visiting b
$q_{b,d}$ ... total load volume before visiting b on day d
$s_{b,d}$ ... service time of block b on day d including waiting times

Borčinova [1] suggests two different formulations of the Capacitated Vehicle Routing Problem building the base of the following formulations.

Mixed Integer Program 9. *minimizing*

$$\sum_{b_1 \in \mathcal{B}^+} \sum_{b_2 \in \mathcal{B}^+} x_{b_1,b_2} \sum_{d \in \mathcal{D}} T_{b_1,b_2,d} + \alpha \sum_{b \in \mathcal{B}'} x_{b_{start},b} \tag{3.34}$$

subject to

$$\sum_{b' \in \mathcal{B}^+} x_{b,b'} = \sum_{b' \in \mathcal{B}^+} x_{b',b} = 1, \qquad \forall b \in \mathcal{B}' \tag{3.35}$$

$$a_{b_1,d} + S_{b_1,d} + T_{b_1,b_2,d} - M(1 - x_{b_1,b_2}) \leq a_{b_2,d}, \qquad \forall b_1, b_2 \in \mathcal{B}', d \in \mathcal{D} \tag{3.36}$$

$$q_{b_1,d} + Q_{b_1,d} - M(1 - x_{b_1,b_2}) \leq q_{b_2,d}, \qquad \forall b_1, b_2 \in \mathcal{B}', d \in \mathcal{D} \tag{3.37}$$

Boolean and non-negativity constraints

$$x_{b_1,b_2} \in \{0, 1\}, \qquad \forall b_1, b_2 \in \mathcal{B}^+ \tag{3.38}$$

$$a_{b,d} \in [T_{min} + T_{0,b,d}, T_{max} - T_{b,N+1,d}], \qquad \forall b \in \mathcal{B}', d \in \mathcal{D} \tag{3.39}$$

$$q_{b,d} \in [0, V - Q_{b,d}], \qquad \forall b \in \mathcal{B}', d \in \mathcal{D} \tag{3.40}$$

The formulation above describes a capacitated VRP with $2D$ resources, i.e. the time and load on every day, and focuses on minimizing the vehicle specific costs VC and the total travel time TT. Arrival time consistency is only achieved by the structure of blocks and not directly taken into account, so also the possibility of waiting times is omitted. Also the driver consistency is not part of the objective function in MIP 9, because using the definition of blocks as described in Definition 3.1.3, leads to a perfect driver consistency.

In Section 5.1, this strict structure will be understood in a softened way, meaning that blocks can be described as set of single stops, instead of whole customers. Especially in a softened setup customers may be served in two or more different blocks. In this case, driver consistency in MIP 9 results just from the structure of blocks, more detailed, from the number of blocks including stops of the regarded customer. For a customer i it holds

$$|\{b \in \mathcal{B}' \ : \ i \in b\}| \geq \sum_{v \in \mathcal{V}} z_{i,v},$$

so the driver consistency of i is limited by the number of blocks including stops of this customer (which is meant by $i \in b$).
Allowing waiting times to improve arrival time consistency leads to the difficulty of measuring the total travel time as the difference between arrival time at the depot and the departure time from depot. But this way of measuring the drivers total travel time results in the great advantage of an MIP formulation, equivalent to MIP 9, which is suitable for every kind of block.

Mixed Integer Program 10. *minimizing*

$$\sum_{b \in \mathcal{B}'} \sum_{d \in \mathcal{D}} a_{b,d}^{end} + \alpha \sum_{b \in \mathcal{B}'} x_{b_{start},b} + \gamma \sum_{i \in \mathcal{N}} l_i \tag{3.41}$$

subject to

$$\sum_{b' \in \mathcal{B}^+ \setminus \{b_{start}, b\}} x_{b,b'} \geq 1, \qquad \forall b \in \mathcal{B}' \tag{3.42}$$

$$\sum_{b' \in \mathcal{B}^+ \setminus \{b_{end}, b\}} x_{b',b} \geq 1, \qquad \forall b \in \mathcal{B}' \tag{3.43}$$

$$\sum_{b' \in \mathcal{B}' \cup \{b_{end}\}} x_{b',b_{start}} = \sum_{b' \in \mathcal{B}' \cup \{b_{start}\}} x_{b_{end},b'} = 0, \tag{3.44}$$

$$\sum_{b' \in \mathcal{B}'} x_{b_{start},b'} = \sum_{b' \in \mathcal{B}'} x_{b',b_{end}} \geq 1, \tag{3.45}$$

$$x_{b_1,b_2} + x_{b_2,b_1} \leq 1, \qquad \forall b_1, b_2 \in \mathcal{B}^+ \tag{3.46}$$

$$x_{b_1,b_2} + x_{b_2,b_3} - 1 \leq x_{b_1,b_3}, \qquad \forall b_1, b_2, b_3 \in \mathcal{B}' \tag{3.47}$$

$$x_{b_1,b_2} + x_{b_1,b_3} - 1 \leq x_{b_2,b_3} + x_{b_3,b_2}, \qquad \forall b_1, b_2, b_3 \in \mathcal{B}' \tag{3.48}$$

$$x_{b_1,b_3} + x_{b_2,b_3} - 1 \leq x_{b_1,b_2} + x_{b_2,b_1}, \qquad \forall b_1, b_2, b_3 \in \mathcal{B}' \tag{3.49}$$

$$a_{b_1,d} + s_{b_1,d} + T_{b_1,b_2,d} - M(1 - x_{b_1,b_2}) \leq a_{b_2,d}, \qquad \forall b_1, b_2 \in \mathcal{B}', d \in \mathcal{D} \tag{3.50}$$

$$a_{b_1,d} + s_{b_1,d} + T_{b_1,b_{end},d} - M(1 - x_{b_1,b_2}) \leq a^{end}_{b_2,d}, \qquad \forall b_1, b_2 \in \mathcal{B}', d \in \mathcal{D} \tag{3.51}$$

$$a_{b,d} + s_{b,d} + T_{b,b_{end},d} \leq a^{end}_{b,d}, \qquad \forall b \in \mathcal{B}', d \in \mathcal{D} \tag{3.52}$$

$$a^{start}_{b_1,d} + T_{b_{start},b_2,d} - M(2 - x_{b_2,b_1} - x_{b_1,b_{end}}) \leq a_{b_2,d}, \qquad \forall b_1, b_2 \in \mathcal{B}', d \in \mathcal{D} \tag{3.53}$$

$$a^{start}_{b,d} + T_{b_{start},b,d} - M(1 - x_{b,b_{end}}) \leq a_{b,d}, \qquad \forall b \in \mathcal{B}', d \in \mathcal{D} \tag{3.54}$$

$$a^{start}_{b,d} \leq a^{end}_{b,d}, \qquad \forall b \in \mathcal{B}', d \in \mathcal{D} \tag{3.55}$$

$$q_{b_1,d} + Q_{b_1,d} - M(1 - x_{b_1,b_2}) \leq q_{b_2,d}, \qquad \forall b_1, b_2 \in \mathcal{B}', d \in \mathcal{D} \tag{3.56}$$

inner block arrival times

$$a_{i,d} + S_{i,d} + T_{i,j} \leq a_{j,d}, \qquad \forall b \in \mathcal{B}', (i,j,d) \in \mathfrak{R}_b \tag{3.57}$$

$$a_{b,d} = a_{\zeta_{b,d},d}, \qquad \forall b \in \mathcal{B}', d \in \mathcal{D} \tag{3.58}$$

$$a_{\eta_{b,d},d} + S_{\eta_b,d} - a_{\zeta_{b,d},d} \leq s_{b,d}, \qquad \forall b \in \mathcal{B}', d \in \mathcal{D} \tag{3.59}$$

Boolean and non-negativity constraints

$$x_{b_1,b_2} \in \{0,1\}, \qquad \forall b_1, b_2 \in \mathcal{B}^+ \tag{3.60}$$

$$a_{b,d} \in [T_{min} + T_{0,b,d}, T_{max} - T_{b,N+1,d}], \qquad \forall b \in \mathcal{B}', d \in \mathcal{D} \tag{3.61}$$

$$q_{b,d} \in [0, V - Q_{b,d}], \qquad \forall b \in \mathcal{B}', d \in \mathcal{D} \tag{3.62}$$

On the first sight, it is not clear that MIP 10 defines a Vehicle Routing Problem. This results from the formulation of routes as strict total orders instead of paths, defined by

$$b_1 \prec b_2 :\Leftrightarrow x_{b_1,b_2} = 1$$
$$b_1 \asymp b_2 :\Leftrightarrow b_1 \prec b_2 \ \vee \ b_2 \prec b_1 \ \vee \ b_1 = b_2$$

Theorem 3.2.4. *Each feasible solution of MIP 10 defines an equivalence relation $\asymp$ on the block set $\mathcal{B}'$.*

Proof. Let $\prec$ and $\asymp$ be defined as above, then the reflexivity and the symmetry of $\asymp$ are given by definition. To proof the transitivity for three pairwise distinct blocks b_1, b_2 and b_3 the following four cases can be distinguished.

$$b_1 \prec b_2 \ \wedge \ b_2 \prec b_3 \overset{3.47}{\Rightarrow} b_1 \prec b_3$$
$$b_1 \prec b_2 \ \wedge \ b_3 \prec b_2 \overset{3.49}{\Rightarrow} b_1 \prec b_3 \ \vee \ b_3 \prec b_1$$
$$b_2 \prec b_1 \ \wedge \ b_2 \prec b_3 \overset{3.48}{\Rightarrow} b_1 \prec b_3 \ \vee \ b_3 \prec b_1$$
$$b_2 \prec b_1 \ \wedge \ b_3 \prec b_2 \overset{3.47}{\Rightarrow} b_3 \prec b_1$$

In all cases, it holds $b_1 \asymp b_3$. □

$\mathcal{B}'$ can be uniquely written as the union of its equivalence classes. Simultaneously $\prec$ defines a strict total order on every $[b]_\succ \in \mathcal{B}'/_\succ$, so the elements of $[b]_\succ$ define a unique ascending chain with a unique maximal and a unique minimal element (see [13]). Constraint (3.43) assumes that the minimal element of $[b]_\succ$ must have a predecessor in the routing graph, which is not in $\mathcal{B}'$. So this minimal element must be visited right after the vehicles departure from the depot. Analogously, Constraint (3.42) gives that the maximal element of $[b]_\succ$ is the last block being visited before the vehicle arrives at the depot block b_{end}. In summary, $\mathcal{B}'_\succ$ is the set of routes from b_{start} to b_{end} visiting all blocks from $\mathcal{B}'$.
Waiting times are realised in MIP 10 by adding additional constraints (3.57) to (3.59) for the inner block arrival times and exchanging $S_{b,d}$ by a continuous variable $s_{b,d}$. In this case, $\mathfrak{R}_b$ is the analogue to $\mathfrak{R}_v$ for blocks defined by

$$\mathfrak{R}_b := \{(i,j,d) \in b \times b \times \mathcal{D} \mid j \text{ is served directly after } i \text{ on day } d \text{ in block } b\}$$

The total travel time is measured by the difference of the continues variable $a^{\text{end}}_{b,d} - a^{\text{start}}_{b,d}$ and minimized in the objective. It holds

$$3.51, 3.52 \quad \rightarrow \quad a^{\text{end}}_{b,d} = \max_{b' \in [b]_\succ} a_{b',d} + s_{b',d} + T_{b',b_{\text{end}},d}$$

$$3.53, 3.54, 3.55 \quad \rightarrow \quad a^{\text{start}}_{b,d} = \begin{cases} \min_{b' \in [b]_\succ} a_{b',d} - T_{b_{\text{start}},b',d} & \text{, if } x_{b,b_{\text{end}}=1} \\ \text{arbitrary in } [0, a^{\text{end}}_{b,d}] & \text{, otherwise} \end{cases}$$

So, for the last blocks of every route the difference $a^{\text{end}}_{b,d} - a^{\text{start}}_{b,d}$ is equal to the total travel time of the driver, and for all other blocks the MIP chooses $a^{\text{start}}_{b,d} = a^{\text{end}}_{b,d}$. The fact that every block is connected to every following block of the route, leads to a correct measuring of the arrival times, even if a block does not contain any customer being served on a specific day. Due to the formulation of the route as a strict total order, this block would be skipped and not visited on this day.
Again driver consistency in MIP 10 is not directly taken into account. Introducing a soft constrained driver consistency measurement, like it is done in MIP 4, would result in $|\mathcal{B}'||\mathcal{V}|$ additional boolean y-variables. Instead, it is also possible using the fact that b_1 and b_2 are visited from two different vehicles if and only if $x_{b_1,b_2} + x_{b_2,b_1} = 0$. Subsequently, the driver consistency of a customer being part of both blocks is increased.

$$DC := \sum_{i \in \mathcal{N}} \omega_i \sum_{\substack{(b_1,b_2) \in \mathcal{B}' \times \mathcal{B}' \\ b_1 \neq b_2 \ \wedge \ i \in b_1 \cap b_2}} (1 - x_{b_1,b_2} - x_{b_2,b_1})$$

Defining $B_i := |\{b \in \mathcal{B}' : i \in b\}|$ it is suggested to choose $\omega_i := \frac{1}{B_i - 1}$ for $B_1 > 2$ and $\omega_i = 0$ otherwise, justified by the following example.

Example 1. *Let $i, j \in \mathcal{N}$ be customers being part of B_i and B_j different blocks, then also the maximal number of drivers being responsible for these customers is lower equal B_i and B_j. Assuming the following two cases*

- *for i holds that all blocks are visited from different drivers*
- *for j holds that all but one block is visited from the same driver*

$$\sum_{\substack{(b_1,b_2)\in\mathcal{B}'\times\mathcal{B}' \\ b_1\neq b_2 \ \wedge \ i\in b_1\cap b_2}} (1 - x_{b_1,b_2} - x_{b_2,b_1}) = B_i(B_i - 1)$$

$$\sum_{\substack{(b_1,b_2)\in\mathcal{B}'\times\mathcal{B}' \\ b_1\neq b_2 \ \wedge \ j\in b_1\cap b_2}} (1 - x_{b_1,b_2} - x_{b_2,b_1}) = 2(B_j - 1)$$

So choosing $\omega_i := \frac{1}{B_i - 1}$ follows that in both cases the value of driver consistency is equal to the number of drivers being responsible for the customer.

In general, it is not possible defining ω_i in a way that the value of driver consistency is equal to the number of different drivers. So, including the driver consistency, as it is defined above, to the objective function does not lead to an equivalent formulation of the MIP 4. In Section 4.3, the results of solving MIP 10 for small instances will be presented.

In summary, this chapter suggests a grouping of customers in form of blocks. Every block has D demands of different resources being related to D different service times. Also the travel time between two blocks may vary between different days. The way these blocks are generated focuses on an equal amount of stops being visited every day, minimizing the differences of service time and demand between several days. Taking also the customer geographically locality into account, leads to more uniform travel times between blocks but also within a block. The result is a VRP where consistency is simply achieved by the structure of blocks. Furthermore the usage of blocks leads to the possibility of formulating a multi objective ConVRP as an MIP including just $|\mathcal{B}'|^2$ boolean variables, giving a large potential of different further solving strategies.

Chapter 4

Numerical Experiments

While most works like [14] suggest random distributed locations to generate instances for testing the procedures, the results of this thesis base on real world data, highlighting the advantages of the combinatorial customer grouping. Section 4.1 describes how the real world instances are created. The capability of block generation is simulated using three different distributions of the number of days the customers require service on. The performance of the combinatorial customer grouping and a comparison to random distributed locations is presented in Section 4.2. Also the key motivation of this thesis, the influence of the customer's position in the routing plan on the arrival time consistency, is justified. Following, the results for small instances are discussed, including a detailed analysis of a specific computed routing plan (see Section 4.3).
The common time horizon considered by most works treating the ConVRP is 5 days (see [4], [14], [18], [19]). So, also the data used and created as part of this thesis consists of customers requiring service on at least one and at most every 5 days.

4.1 Generation of Real World Instances

Beside the degree of practical relevance, one key difference between randomly created instances and real world instances is the distribution of the customer's location. While most created instances suggest a uniform generation of customer locations ([14]), the real world case is that locations accumulate in many different centres like cities, villages or districts. This means that customers can be more easily clustered, which also applies for the combinatorial customer grouping.
For the creation of real world instances, the GPS coordinates of customers of a small package delivery company have been used. Based on this coordinates a cost matrix has been computed giving the exact travel times, distances and energy demands under real world conditions between every pair of locations. Afterwords the coordinates have been anonymized by using a randomly chosen isometry. This way, also the spherical distances between two GPS locations remain the same.
Using this real world data, problem instances to four different sizes are generated by picking 20, 50, 70 and 100 customers out of a pool of roundly 2000 locations. The

amount of stops varies with the mean number of days a customer requires service. Groër et al. [14] indicates a daily service probability between 60% and 90%, meaning that at least six out of ten customers are complete. A high amount of complete customers significantly simplifies the problems appearing with regards on consistency. This thesis clearly focuses on the difficulty of a high diversity of customers delivery specifications and suggests three different distribution types for the amount of delivery days (see Figure 4.1).

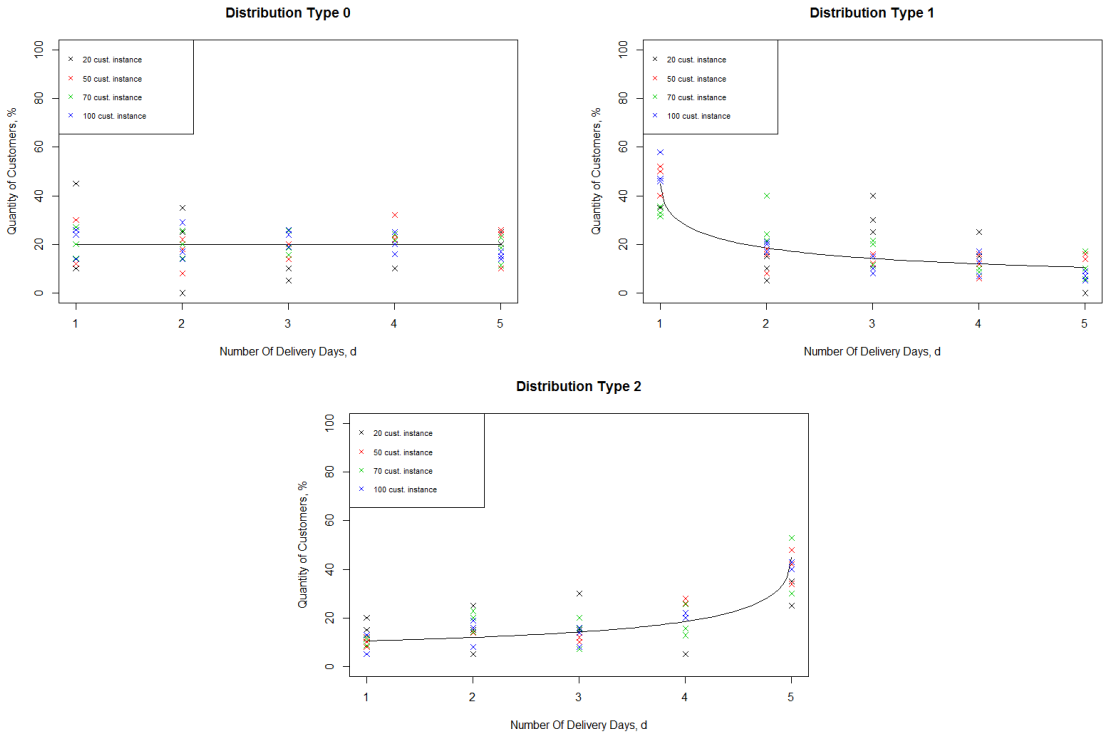


Figure 4.1: Different distribution types of the number of delivery days of a customer

Distribution 0 is a uniform distribution. Distribution 1 is the distribution resulting from the random variable $\lfloor X^2 \cdot D + 1 \rfloor$, where X is the uniformly distributed continuous random variable on the interval $(0, 1)$. Customers being generated along this distribution are likely to require service on a few number of days. Conversely, distribution 2 results from the random variable above reflected through $\frac{D+1}{2}$, leading to a higher probability for complete customers.

Theorem 4.1.1. *Let $X \in (0,1)$ be an uniformly distributed random variable, $p \in \mathbb{N}$, $d \in \{1, ..., D\}$ and $Y_p := \lfloor X^p \cdot D + 1 \rfloor$, then*

$$Y_p \in 1, ..., D$$

$$\mathbb{P}(Y_p = d) = \sqrt[p]{\frac{d}{D}} - \sqrt[p]{\frac{d-1}{D}}$$

	$d=1$	$d=2$	$d=3$	$d=4$	$d=5$
Distribution 0	$20,00\%$	$20,00\%$	$20,00\%$	$20,00\%$	$20,00\%$
Distribution 1	$44,72\%$	$18,52\%$	$14,21\%$	$11,98\%$	$10,56\%$
Distribution 2	$10,56\%$	$11,98\%$	$14,21\%$	$18,52\%$	$44,72\%$

Table 4.1: Possibilities of the amount of delivery days

Proof. With X also X^p is a random variable in $(0,1)$ and therefore

$$X^p \cdot D + 1 \in (1, D+1) \Rightarrow Y_p := \lfloor X^p \cdot D + 1 \rfloor \in \{1, ..., D\}.$$

Given $d \in \{1, ..., D\}$, then $Y_p = d$ is equivalent to

$$\begin{aligned} d \leq X^p \cdot D + 1 &< d+1 \\ \sqrt[p]{\frac{d-1}{D}} \leq \quad X \quad &< \sqrt[p]{\frac{d}{D}}, \end{aligned}$$

resulting in $\mathbb{P}(Y_p = d) = \sqrt[p]{\frac{d}{D}} - \sqrt[p]{\frac{d-1}{D}}$. □

Remark 4.1.2. *The distribution resulting from the random variable Y_p reflected through $\frac{D+1}{2}$, i.e. $\overline{Y}_p := D + 1 - Y$, subsequently has a possibility of*

$$\mathbb{P}(\overline{Y}_p = d) = \sqrt[p]{\frac{D+1-d}{D}} - \sqrt[p]{\frac{D-d}{D}}.$$

Assuming $D = 5$ and $p = 2$ results in the distributions and possibilities highlighted in Figure 4.1 and Table 4.1. The highest percentage of complete customers is achieved by instances being generated according distribution 2, guaranteeing a roundly 4.5 higher probability of a daily service requirement than of just one delivery day. Still this probability is chosen notably lower than the percentages suggested by [14].
The number of delivery days alone does not define the delivery specifications of a customer. The actual days, a customer requires service on, are also chosen randomly. For a customer requiring service on d days there are $\binom{D}{d}$ combinations of delivery possibilities, which are assumed to be uniformly distributed. Thinking about B2B delivery companies, it may be from practical interest weighting these combinations with different possibility values, for example if most companies do not want to be supplied on Fridays. In this case, it may be more sensible calculating a routing for the remaining days of the time horizon. Therefore, this exception is not taken into account at the generation of problem instances.
To every different number of customers and every distribution type three different instances are generated, indicated by an $id \in \{0, 1, 2\}$. A detailed summary of all instance specifications, including customer numbers to different types and data regarding the results of the combinatorial customer grouping in complete blocks, can be find in Table

4.2. The first three columns indicate the instance type. As defined before, complete customers are customers requiring service on every day of the time horizon. These customers are treated as complete blocks. The number of frequent customers specifies how many customers require service on more than one day. Column 6 describes the number of customers assigned to blocks by the customer grouping heuristic. Complete customers are included in the last two values, i.e. to frequent customers and customers assigned to blocks. The number of generated complete blocks and the total height of all blocks added up are two parameters describing the performance of the customer grouping algorithm. The last three columns refer to the distribution of customers per day.

After the generation of delivery stops, the customer service requirements have to be

NumC	DistrT	InstID	complC	frequC	inBlocks	NumB	totalBlockH	minCPD	maxCPD	meanCPD
20	0	0	5	14	9	7	7	11	16	12.6
20	0	1	3	11	9	6	6	6	11	9
20	0	2	4	18	15	8	10	10	14	12.4
50	0	0	5	35	37	18	19	22	31	26.2
50	0	1	13	43	39	23	28	28	36	32.8
50	0	2	12	44	38	21	29	32	41	34.8
70	0	0	13	56	55	28	35	35	45	40.8
70	0	1	16	60	59	32	40	40	50	45
70	0	2	8	51	67	32	38	38	40	38.6
100	0	0	14	74	85	44	51	51	60	56.4
100	0	1	17	86	87	47	53	53	63	58.6
100	0	2	15	76	89	46	54	54	65	58
20	1	0	0	13	12	4	6	6	12	9.4
20	1	1	2	12	8	4	5	5	13	9.4
20	1	2	1	13	14	6	8	9	12	10.4
50	1	0	7	25	37	19	21	21	26	23.6
50	1	1	7	24	44	20	20	20	24	21.2
50	1	2	8	30	36	20	22	22	30	24.8
70	1	0	12	48	57	32	33	33	41	36.4
70	1	1	7	45	44	23	23	23	40	32.6
70	1	2	4	47	54	23	26	26	34	30.2
100	1	0	5	42	83	33	32	33	41	36.2
100	1	1	9	54	83	40	41	41	50	44.4
100	1	2	7	53	79	39	39	39	48	43.2
20	2	0	7	16	10	8	9	10	17	13.4
20	2	1	8	16	12	9	10	11	16	13.6
20	2	2	5	17	12	7	9	9	16	12.6
50	2	0	24	44	47	33	37	37	40	38.4
50	2	1	21	45	45	31	35	35	39	37.6
50	2	2	17	46	39	26	30	33	41	36
70	2	0	37	61	61	47	49	49	57	53
70	2	1	21	64	60	35	44	44	54	48.8
70	2	2	21	62	58	36	41	41	51	46.2
100	2	0	43	87	91	62	70	70	78	74.8
100	2	1	40	95	84	56	67	68	82	74.2
100	2	2	43	87	87	60	67	67	76	72.8

Table 4.2: Instance specifications including data regarding the generated blocks
columns from left to right: number of customers, distribution type, instance ID, complete customers, frequent customers, customers in blocks, number of complete blocks, total block height, minimum of customers per day, maximum of customers per day, mean of customers per day

furnished with service times and demands. For every customer a basic service time between 3 and 15 minutes and a basic demand of 1 to 10 kilogram are chosen by using a uniformly distributed random variable. With these two values the actual service times and demands at every stop are calculated, by firstly deciding with equal probability if the basic values should be increased or decreased. This way guarantees that a higher demand tendentially means a higher service time. The second step is choosing randomly the amount of change, whereby the deviation is limited to 10% of the basic service time and 25% of the basic demand.

4.2 Performance of the Combinatorial Customer Grouping

The ideal case of the preprocessing would be that all customers are assigned to a complete block and just a routing between blocks has to be performed, meaning that the dimension of time may almost completely be ignored. This assumes that on every day the same amount of customers require service. In practise, there will be a non empty set of remaining customers which are not assigned to a complete block. The size of this set is highly affected by the distribution of deliveries and also by the shape of delivery requirements over the whole time horizon of every single customer. Table 4.3 and Figure 4.2 show the impact of the number of customers, distribution type and the number of deliveries per day on the number of customers assigned to complete blocks.

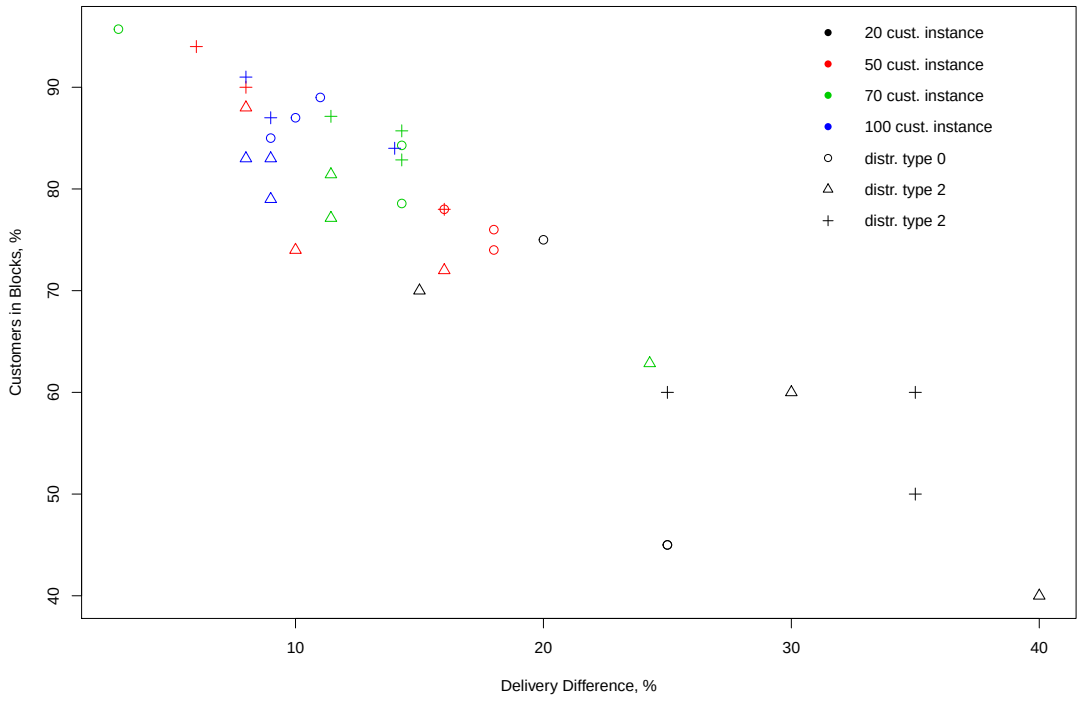


Figure 4.2: Dependency between the percentage of customers in blocks and the largest difference of deliveries amount per day

Figure 4.2 shows a linear indirect proportionality between the number of customer assigned to complete blocks and the largest difference of deliveries amount per day, measured by $\frac{(maxCPD-minCPD)}{NumC}$. This results from the fact that $|\mathcal{B}' \cap \mathcal{B}^C| \leq minCPD$, where $\mathcal{B}'$ is still the set of blocks computed in Section 3.1.3 and highlighted in Remark 3.2.1. It even holds that $|\mathcal{B}' \cap \mathcal{B}^C| \leq \sum_{b \in \mathcal{B}' \cap \mathcal{B}^C} h_b \leq minCPD$. Therefore, the total height of all blocks is an important value to measure the performance of the combinatorial customer grouping. The mean value of $minCPD - totalBlockH$ over all instances

DistrT \ NumC	20	50	70	100	mean
0	55,00%	76,00%	86,19%	87,00%	76,05%
1	56,67%	78,00%	73,81%	81,67%	72,54%
2	56,67%	87,33%	85,24%	87,33%	79,14%
mean	56,11%	80,44%	81,75%	85,33%	75,91%

Table 4.3: Average percentages of customers assigned to complete blocks

is 0.5, regarding instances with 70 and more customers this value decreases to 0.11. At 75% of all instances, the maximum value $minCPD = totalBlockH$ is reached.
Regarding the distribution type a slightly worse value of customers in blocks can be seen at the instances from distribution type 1. This may result from the higher probability of one and two day customers. The conjecture is that if several customer, requiring service on a few number of days, are supplied on the same day, than the value of $maxCPD - minCPD$ increases and this value affects the number of customers in blocks (see Figure 4.2).
A more significant impact on the number of customers in blocks has the size of the instance (see Table 4.3). While instances consisting of 20 customers only have round 56% customers in blocks, instances with 50 customers and more achieve values above 80%.
The other way considering the performance of the combinatorial customer grouping is the locally point of view. In practice, the ideal case that only closely located customers are matched to a complete block constitute an exception. Proleptically it can be said that greater instances achieve better results than smaller ones. The closeness of customers in a block is always measured by the distance to the median centre of a block (compare with MIP 6).
As it can be seen in Section 3.1, the grouping of customers is highly influenced by their delivery days instead of their location. It is possible that two customers, lying close to each other, may not be part of the same block, because they require service on the same days. Conversely, it can also happen that a block is created by matching two customers with a large distance between them, because of a lack of other possible combinations. Working with real world data shows, the chance that the customers of a single block are lying close to each other is significantly higher than if uniformly distributed data is used.
Giving comparability between real world and uniformly distributed locations the customers of every problem instance are furnished with uniformly distributed GPS coor-

dinates within the rectangle $[minLat, maxLat] \times [minLon, maxLon]$. Afterwards the customer grouping is performed again for every instance and the distance of every customer to the blocks median centre is measured. The problem instances with 20 customers show no significant difference between real world and uniform data (see Figure 4.3). Conversely, it can be seen in Figure 4.4 that instances with 100 customers achieve clearly smaller values when real world data is used. Even though the considered region is rural, median values below 20 kilometres are achieved. Instances `cust100_distr1_id2` and `cust100_distr2_id0` actually have a median distance to the blocks geometric centres of less than 10 kilometres.

4.3 Resulting Routes

The MIPs introduced in Section 3.2.1 were implemented in Java and solved by IBMs CPLEX solver on an ACER Aspire 5750G equipped with Intel Core i3 CPU of 2.1 GHz. In detail, solutions of MIP 10 were computed for all instances consisting of 20 customers, whereby only the complete blocks were taken into account. The amount of the generated complete blocks per instance is written in the 7th column of Table 4.2. The weights in the objective function are chosen as follows, in the sake of getting a comparability between the different components.

$$\alpha = 36000$$
$$\gamma = 1$$

The total travel time TT and the arrival time consistency AC are given in seconds, so the vehicle costs of one vehicle is penalized with the maximal allowed operation time of the vehicle, 10 hours in seconds. As mentioned in Section 3.2.1, driver consistency has not to be taken into account as long all customers are part of a single block.
Table 4.4 summarizes the achieved results of the performed computing. Getting an overview on how arrival time consistency effects the routing efficiency of a solution, MIP 10 is again solved with $\gamma = 0$ for every instance. The differing values of the total travel time, including AC and without taking AC into account, and the percentage of the regarding increase are described in the columns 2-4 in Table 4.4. For every customer in the instance, being served on more than one day, the interval length of the arrival times is computed. The minimal, maximal and mean values being achieved for every instance are summarized in the columns 5-7. Also important to say is that all routings, except of instance `cust20_distr0_id2`, are fulfilled by a single vehicle. Additionally, instance `cust20_distr0_id2` is discussed in detail in Section 4.3.1. The disadvantages of the combinatorial customer grouping for small instances are discussed in Section 4.2. Especially the influence of the instance size and also the distribution type on the distance to the blocks geometric median (Figure 4.3 and 4.4) leads to a similar notice regarding the arrival time consistency. Clearly a good arrival time consistency is difficult to achieve if the inner block geographical locality is not given. Still the mean maximal length of the customers arrival time interval over all instances is 11min 33s, simultaneously the impact of arrival time consistency on the total travel is - with a mean increase of the

InstName	TT	TT wAC	TT Perc	min AC	max AC	mean AC
cust20_distr0_id0	33h 56min 9s	33h 42min 6s	0.69	0min 0s	1min 0s	0min 17s
cust20_distr0_id1	33h 26min 38s	32h 45min 46s	2.08	0min 0s	1min 0s	0min 17s
cust20_distr0_id2	49h 44min 50s	47h 59min 12s	3.67	0min 0s	51min 2s	7min 42s
cust20_distr1_id0	37h 47min 26s	35h 58min 47s	5.03	0min 0s	100min 1s	29min 0s
cust20_distr1_id1	32h 37min 50s	31h 30min 38s	3.55	0min 0s	39min 7s	7min 1s
cust20_distr1_id2	41h 23min 11s	37h 50min 15s	9.38	0min 0s	48min 42s	8min 22s
cust20_distr2_id0	41h 41min 10s	41h 21min 16s	0.8	0min 0s	34min 8s	3min 49s
cust20_distr2_id1	45h 18min 15s	43h 40min 16s	3.74	0min 0s	12min 46s	1min 37s
cust20_distr2_id2	43h 21min 14s	43h 15min 1s	0.24	0min 0s	171min 5s	45min 49s

Table 4.4: Arrival time consistency data
columns from left to right: name of the instance, total travel time, total travel time without arrival time consistency, percentage of travel time increase, minimal arrival time difference, maximal arrival time difference, average arrival time difference

total travel time of just 3.24% - comparatively small.
Beside the geographical position of the customers, also the position in the route is a key impact factor on the arrival time consistency. Measuring this correlation, the maximal position difference of every customer within a route is computed. This is done for every computed routing plan of the 9 instances. Subsequently, the resulting values are associated with the customers maximal arrival time difference and viewed in form of a boxplot (Figure 4.5). The flexible departure times from the start depot are influencing the arrival time consistency in a positive way (Figure 4.5 left). Getting rid of this impact, all routes are additionally assumed to start at the same time and the new resulting arrival time differences are again compared in the boxplot on the right side (Figure 4.5).

4.3.1 Example Route

As mentioned previously, instance `cust20_distr0_id2` is the only one requiring two vehicles serving all customers. It is also the instance with the highest value of customers in blocks (Table 4.2). 15 out of 20 customers are part of 8 generated complete blocks. Figure 4.3 shows that the maximal distance to the blocks artificial centre is roundly 20km, meaning that the inner blocks locality is good, but not perfect, making this instance an interesting example.
The generation of complete blocks and the solving of MIP 10 results in the regarding routing plan of blocks, which can be seen in Figure 4.6. The grey number on the left side of the blocks marks the block id. Analogously, the stops of the routes pictured in Figure 4.6 are signed with this block id.
The routing plan in Figure 4.6 is read from the bottom to the top, meaning that customer 16 is the first to be served by vehicle 2 on Monday and so on. Again it can be seen that no customer is separated on two blocks, therefore especially not on two different vehicles. So a perfect driver consistency is the result.
Comparing the blocks with the routes in Figure 4.6 shows that especially block 4 consists of three customers 6, 15 and 16 with highly varying GPS position. This has a significant

impact on the routes and the order in which the customers are served. Also the distance between the customers 13 and 14 in block 1 is large, but its influence on the routing is comparatively smaller.
Focusing on the arrival time consistency of the customers, Figure 4.7 gives an overview on when a stop is visited every day. One key recognition is that the inner blocks locality does not necessarily have an impact on the arrival time consistency. All but one customer of the blocks 1 and 4 achieve perfect arrival time consistency. This results on the one hand from the flexible departure times from the depot. But on the other hand also from the completeness of the blocks, guaranteeing that, even if the locality is not given, the amount of customers being served every day and the customer's position in the routing plan are roughly remaining the same.

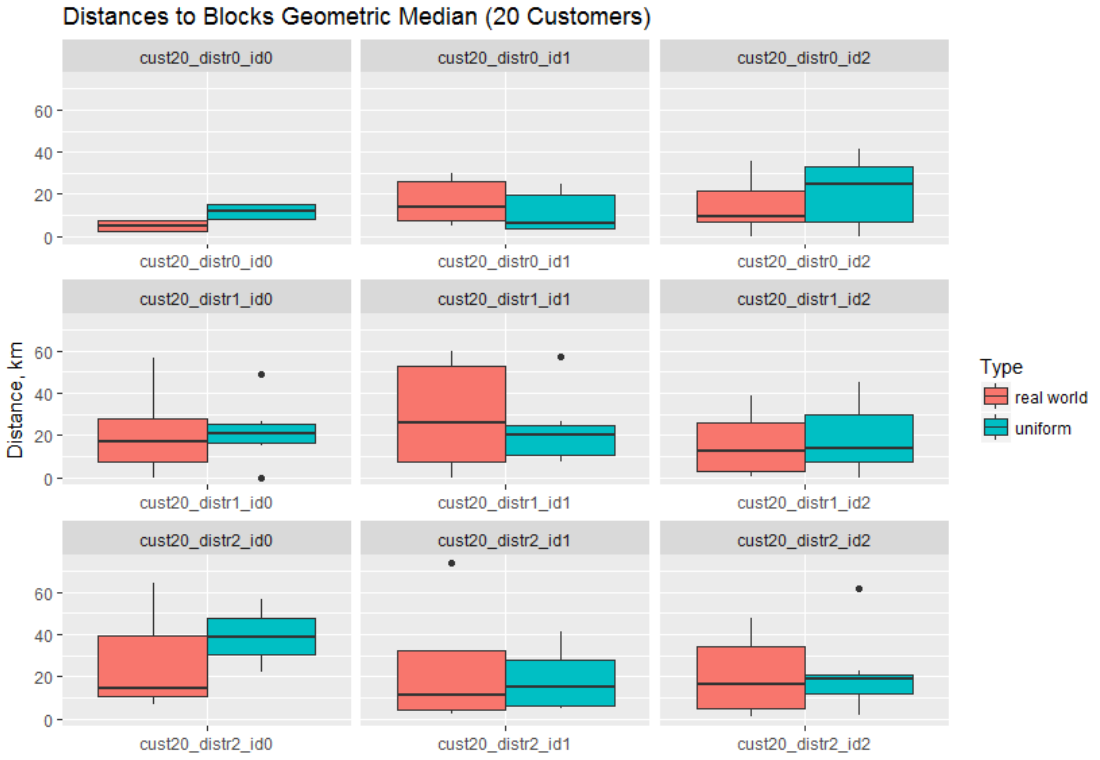


Figure 4.3: Difference in customer grouping between real world and uniformly distributed locations

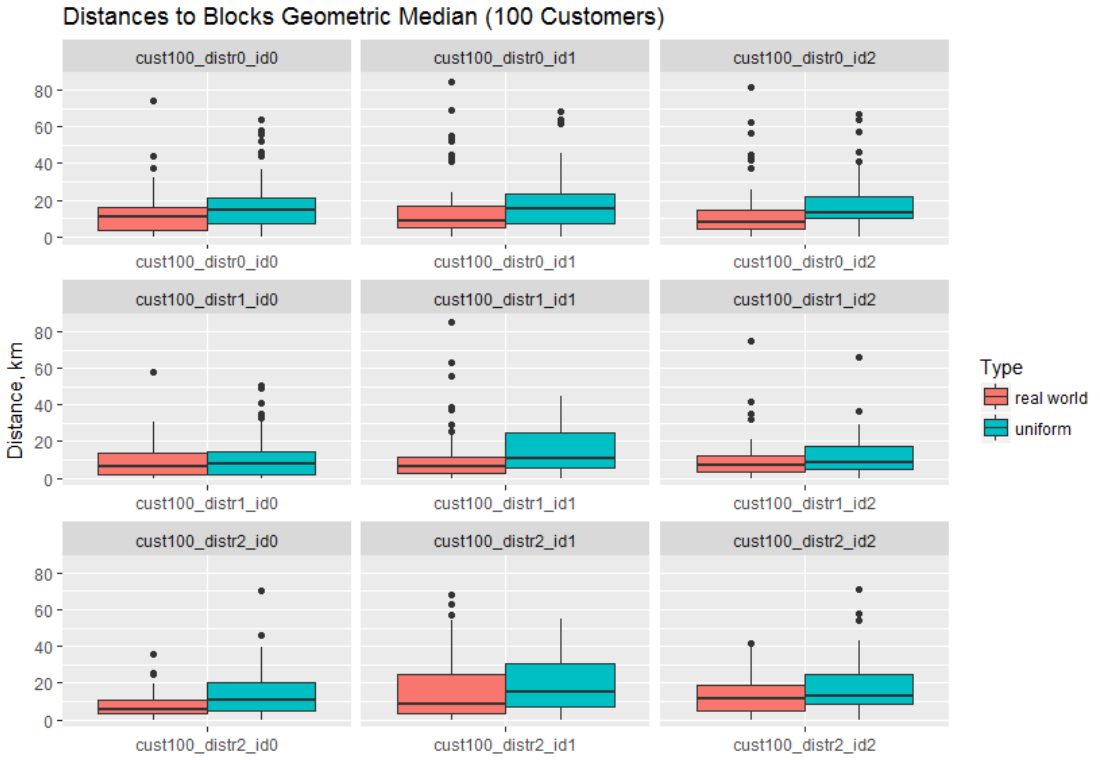


Figure 4.4: Difference in customer grouping between real world and uniformly distributed locations

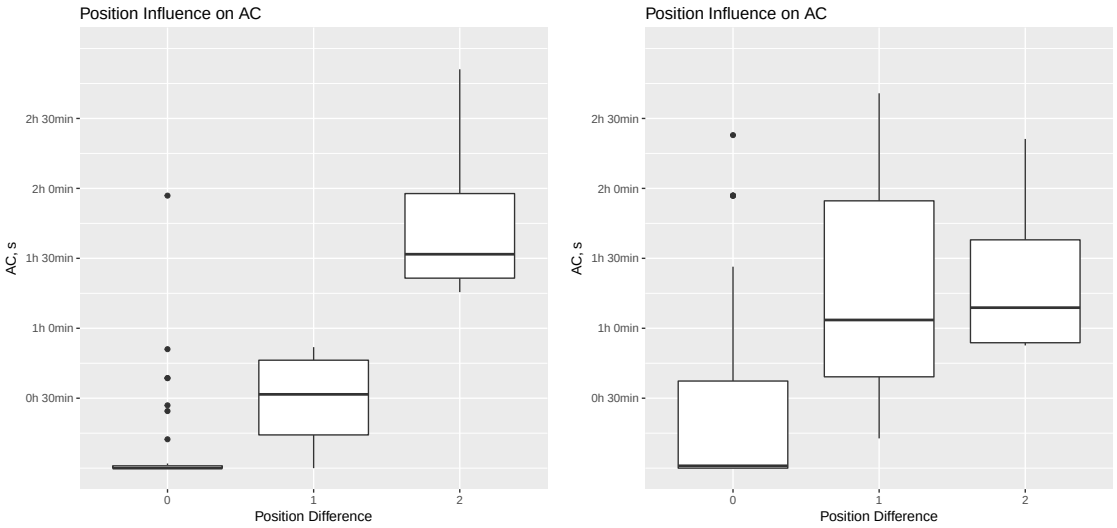


Figure 4.5: Impact of the customer's position in the routing plan on its arrival time consistency with flexible departure times from the depot (left) and without (right)

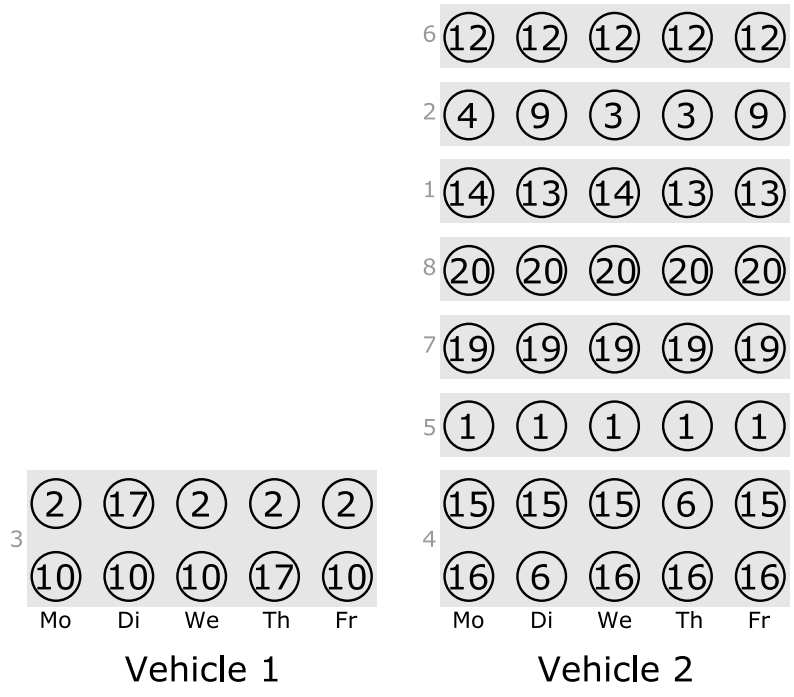
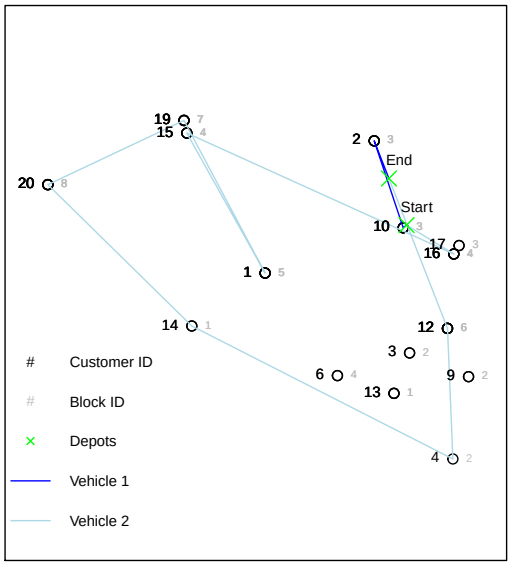
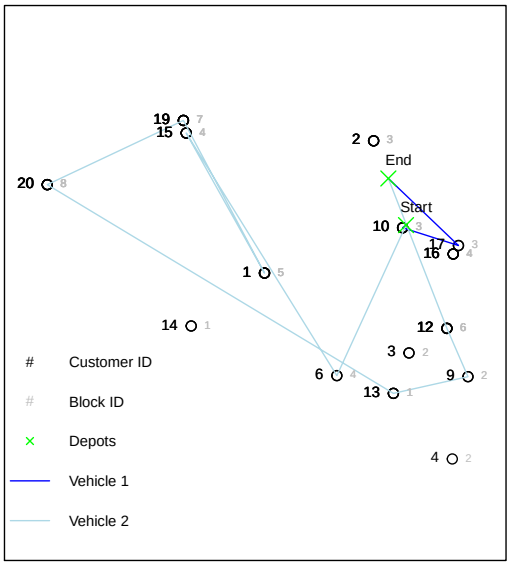


Figure 4.6: Routing plan of blocks

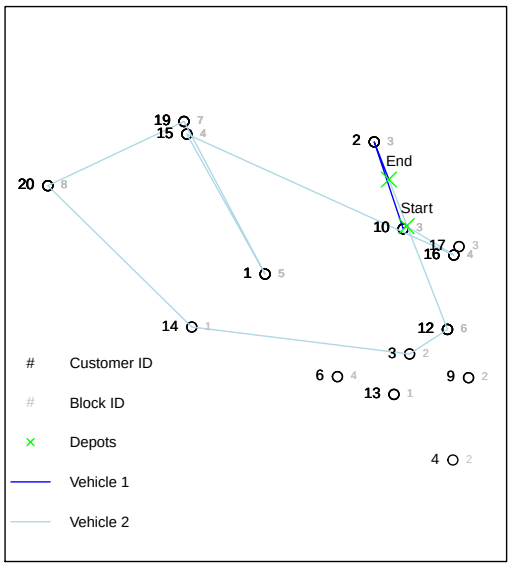
Mo



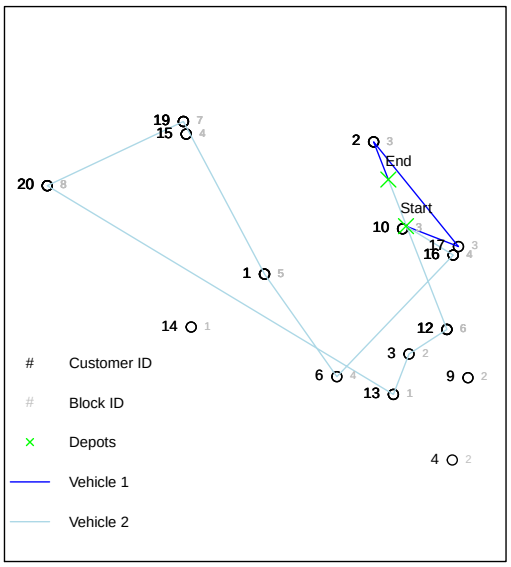
Tu



We



Th



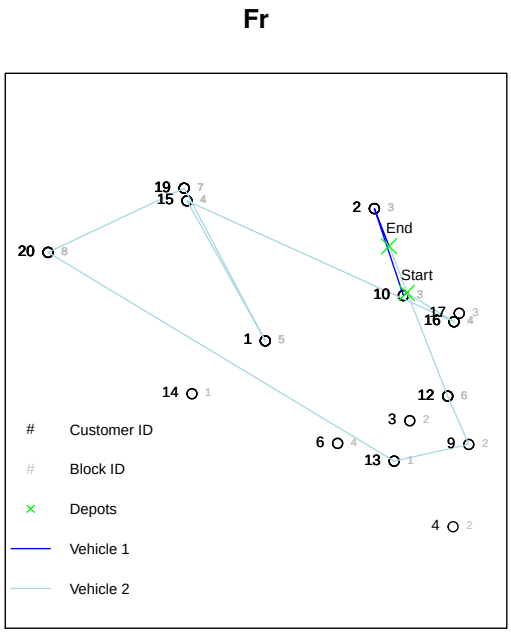


Figure 4.6: Example Daily Routes

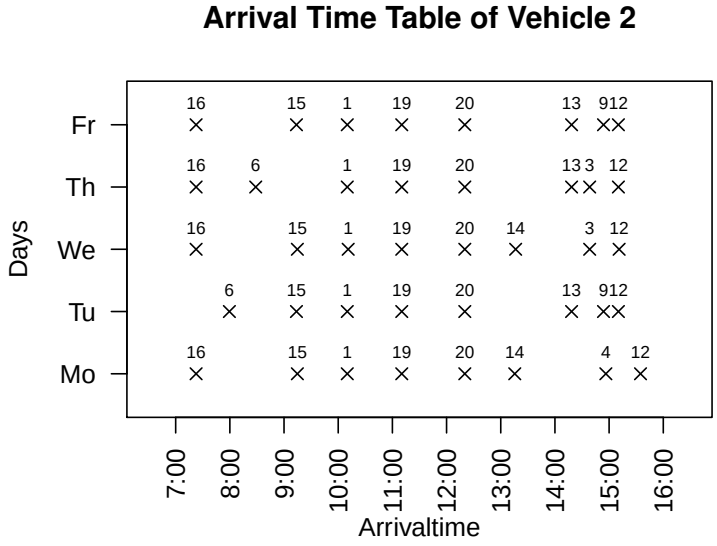


Figure 4.7: Arrival times of the supplied customers

Chapter 5

Conclusion and Outlook

In summary, it can be said that a combinatorial grouping of customers is a strategy worth refining to tackle VRPs with consistency requirements. Both, driver and arrival time consistency, is handled by the structure of the blocks. Whereby the number of different drivers being responsible for a customer is simply limited to the number of blocks the customer is divided in. Also the influence of the combinatorial structures on the arrival time consistency can be proofed.
Even if the inner block geographical position of the customers highly varies, the constance of the customer's position in the routing plan is one key impact factor on the arrival time consistency. This underlines also the motivation behind this approach and shows that the defined combinatorial structure of complete blocks fulfils the right assumptions.
Beside guaranteeing an uniform order of the customers, so that the customer's position in the routing plan remains nearly the same over the regarded time horizon, blocks also give the flexibility for further usage.
In times of Big Data exact methods, especially the solving with mixed integer programs, are cut out by heuristical approaches. Here the structure of blocks can lead its second trump of its high adaptability.

5.1 Further Solving Strategies

As seen in Section 3.2.1, the usage of blocks allow the possibility of formulating a ConVRP as a capacitated VRP with $2D$ resources. So a large range of algorithms fitted on capacitated VRPs is available to solve the ConVRP (see [25]). One key difficulty is the well-definedness of the distance or the time spent to travel between two blocks, if one block does not require service on the specific day. In Section 3.2.1, this is solved by considering the travel time not only between two coincident blocks, but between every pair of blocks within a route. Besides, many heuristics and also exact approaches, like the Branch and Price Algorithm, are valuing a feasible solution with a priori given routes, where the problem of a well-defined distance between two blocks does not occur.

Branch and Price on Blocks

As said, an advantage of Branch and Price algorithms is that the solution specific costs can be calculated under the assumption that the routes are a priori given. Due to the fact that waiting in the sake of improving arrival time consistency is allowed, costs of travel time TT and arrival time consistency AC can not directly be followed from the order in which the customers are supplied, contrary to VC and DC. But it is possible to reduce MIP 4 to an LP computing the exact costs of a assumed feasible solution, like it is done by LP 8 in Section 3.1.4. One main part of the Branch and Price algorithm is the column generation. Feillet et al. [10] suggests a label correcting algorithm to generate new solutions with reduced costs. Its time complexity is said to be highly depending on the tightness of resource limitations. Regarding the soft constrained block variant of the ConVRP the restrictions are not tight enough to solve large instances. The other disadvantage is that the column generation algorithm in [10] also assumes well defined distances between the nodes of the graph. So a block based Branch and Price algorithm, allowing arbitrary blocks, will have to use a different kind of column generation.

Block based LNS

As mentioned in Section 2.2.2, the Large Neighbourhood Search is an often used heuristic, computing nearly optimally feasible solutions, based on destroy and repair operators [23]. Thinking of a block based LNS while comparing the used operators in works like the one of Kovacs et al. [19], it is easily imaginable that a wide variety of block destroying and repairing operators can be defined.
Blocks may be divided into two blocks improving the geographical locality within each block and inserting the resulting blocks at different places in the routing plan. It can also make sense to merge two blocks into one, for example to decrease the amount of drivers being responsible for one customer. Optimal parts of a solution can be fixed by defining big blocks and even an exchange of single stops between two blocks is thinkable. In the case of hard constrained consistency limitations, an adaptation of the operators or a feasibility check of the solution would have to be included. For example a hard constrained driver consistency, measured by the amount of drivers being responsible for a customer, is simply achieved by restricting the number of blocks in which a customer can be divided.

In conclusion, the multiplicity in working with blocks is not limited, especially if consistency is realized in a soft constrained way. These two examples of further usage and especially the whole thesis should motivate the reader for own attempts in using blocks in Vehicle Routing Problems.

Appendix A

Combinatorial Results on Blocks

The difference between block type and block and between customer type and customer is only important regarding the implementation and can be translated in the combinatorial terminology of *labelled* and *unlabelled* objects.

Definition A.0.1 (binary type)**.** Every customer type $[i]_{\simeq} \in \mathcal{N}/_{\simeq}$ can be explicitly decoded in a binary number of D digits, in particular $P_{i,D}$ ... $P_{i,1}$. Converting this binary number in a decimal number results in an integer value between 1 and $2^D - 1$, where $2^D - 1$ stands for the customer type requiring service on every day of the time horizon. This number is called *binary type* $\beta_i := \sum_{d=1}^{D} P_{i,d} 2^{d-1}$.

The uniqueness of the binary representation of an integer also gives the uniqueness of the binary type of a customer type. By definition, the binary type β_i remains constant for each customer in the equivalent class $[i]_{\simeq}$. This guarantees the well-definedness of β_i and shows that the uniqueness only holds for customer types.

Theorem A.0.2. *Two customers $i, j \in \mathcal{N}$ form a complete block $b := \{i, j\} \in \mathcal{B}^C$ of height $H_b = 1$, if and only if the sum of their binary types is equal $2^D - 1$.*

Proof. If $b = i, j \in \mathcal{B}^C$ is a complete block type with height H_b then by Definition 3.1.3 the condition $P_{i,d} + P_{j,d} = H_{b,d} = H_b = 1$ holds. Subsequently,

$$\beta_i + \beta_j = \sum_{d=1}^{D} P_{i,d} 2^{d-1} + \sum_{d=1}^{D} P_{j,d} 2^{d-1} = \sum_{d=1}^{D} (P_{i,d} + P_{j,d}) 2^{d-1} = \sum_{d=1}^{D} 2^{d-1} = 2^D - 1.$$

Conversely, if $\beta_i + \beta_j = 2^D - 1 = \sum_{d=1}^{D} 2^{d-1}$ then

$$\beta_j = \sum_{d=1}^{D} 2^{d-1} - \beta_i = \sum_{d=1}^{D} (1 - P_{i,d}) 2^{d-1}.$$

The uniqueness of the binary representation of an integer number gives $1 = P_{i,d} + P_{j,d} = H_{b,d}$ for every $d \in \mathcal{D}$. This means that $b := \{i, j\}$ is a complete block with height $H_b = \max_{d \in \mathcal{D}} H_{b,d} = 1$. □

The reverse step in the proof above shows that the condition $\sum_{i \in b} \beta_i = 2^D - 1$ is only sufficient, if the number of customers in block b is equal 2.

Example 2. *If we imagine a time horizon of $D = 3$ and three customers with binary types $\beta_1 = 2$, $\beta_2 = 2$ and $\beta_3 = 3$, then it holds $\sum_{i=1}^{3} \beta_i = 2^D - 1 = 7$. But the block $b := \{1, 2, 3\}$ consisting of these three customer types is of the form $(H_{b,1}, H_{b,2}, H_{b,3}) = (1, 3, 0)$ and therefore not complete.*

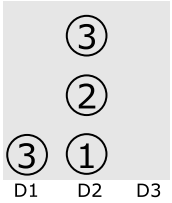


Figure A.1: Example block

But it is possible to formulate a necessary condition for a complete block.

Lemma A.0.3. *Assuming a time horizon of D days, a complete block $b \in \mathcal{B}^C$ of height H_b, then the sum of the binary types holds*

$$\sum_{i \in b} \beta_i = H_b(2^D - 1).$$

Coming back to two customers forming a block of height 1 leads to the following definitions.

Definition A.0.4 (complement customer type)**.** Given a customer type $[i]_\simeq \in \mathcal{N}/_\simeq$ with binary type β_i then the *complement customer type* $[i]^c_\simeq$ is the customer type defined by the binary type $2^D - 1 - \beta_i$ or equivalent

$$[i]^c_\simeq := \{j \in \mathcal{N} \ : \ P_{j,d} = 1 - P_{i,d} \ \forall d \in \mathcal{D}\}$$

Remark A.0.5. *Important to mention is that a representative customer i^c in $\mathcal{N}$ may not exist. In this case, the complement customer type is empty, i.d. $[i]^c_\simeq = \emptyset$.*

Definition A.0.6 (complement block type)**.** Let b be a block fulfilling

$$|[i]^c_\simeq| \geq |[i]_\simeq \cap b| = N_{b,i}, \ \forall i \in b, \tag{A.1}$$

then there exists an injective function $\pi : b \to \mathcal{N}$ with $\pi(i) \in [i]^c_\simeq$ and the *complement block type* to $[b]_\simeq$ is defined by

$$[b]^c_\simeq := [\pi(b)]_\simeq.$$

Remark A.0.7. *The definition of the complement block type is independent from the choice of the injection. Just the existence of π is required, so again it is possible that no complement block type of $[b]_\simeq$ exists.*

The next lemma shows that Condition (A.1) is not only sufficient for the existence of a complement block type, but it is also necessary.

Lemma A.0.8. *Given an arbitrary block b, then there exists a complement block type $[b]^c_\simeq$ if and only if Condition A.1 is fulfilled.*

Proof. Definition 3.1.19 shows that Condition A.1 even holds for every block in $[b]_\simeq$ and therefore also $|[i]^c_\simeq| \geq N_{b,i}$ for every $i \in b$. This leads per definition to the existence of a complement block type.
Conversely, assuming $|[i]^c_\simeq| < N_{b,i}$ for an arbitrary $i \in b$, Theorem 3.1.20 shows that also $|[i]^c_\simeq| < N_{b^*,i}$ for every other block $b^* \in [b]_\simeq$ is fulfilled. Given $i^* \in b^* \cap [i]_\simeq$, then a function $\pi : b^* \to \mathcal{N}$ with $\pi(i^*) \in [i]^c_\simeq$ clearly can never be injective. □

Corollary A.0.9. *Let $b_1 \in [b]_\simeq$ and $b_2 \in [b]^c_\simeq$ then there even exists a bijection $\overline{\pi} : b_1 \to b_2$ with $\overline{\pi}(i) \in [i]^c_\simeq$.*

Proof. Let $b' := \pi(b)$ where π is the injection in Definition A.0.6. Choosing $\pi_1 \in \mathfrak{Bij}(b_1, b)$ and $\pi_2 \in \mathfrak{Bij}(b', b_2)$ as the bijections given by Definition 3.1.8 and defining $\overline{\pi} := \pi_2 \circ \pi \circ \pi_1$ shows the statement. □

In the following sections, exceptions like $[i]^c_\simeq \notin \mathcal{N}/_\simeq$ and $[b]^c_\simeq \notin \mathcal{B}/_\simeq$ are not taken into account. Under this assumption the next three statements are direct consequences of the results above.

Lemma A.0.10. *Assuming two block types $[b_1]_\simeq, [b_2]_\simeq \in \mathcal{B}/_\simeq$, if $[b_1]_\simeq$ is a sub block type of $[b_2]_\simeq$, then $[b_1]^c_\simeq$ is a sub block type of $[b_2]^c_\simeq$, i.e.*

$$[b_1]_\simeq \leq [b_2]_\simeq \Leftrightarrow [b_1]^c_\simeq \leq [b_2]^c_\simeq$$

Corollary A.0.11. *A block type $[b]_\simeq \in \mathcal{B}/_\simeq$ has no sub blocks if and only if $[b]^c_\simeq$ has no sub blocks and therefore*

$$[b]_\simeq \in \mathcal{B}^U/_\simeq \Leftrightarrow [b]^c_\simeq \in \mathcal{B}^U/_\simeq$$

Theorem A.0.12. *Given an indivisible, complete block type $[b]_\simeq \in \mathcal{B}^U/_\simeq$ then $[b]_\simeq$ is its own complement block type, if and only if it is a block of height 1 consisting of two matching customers, i.e.*

$$[b]_\simeq = [b]^c_\simeq \Leftrightarrow b = \{i, j\} : [i]^c_\simeq = [j]_\simeq$$

Proof. Every block type consisting of two matching customers is complete, indivisible and trivially the complement block type of itself. Reversely if a block type $[b]_\simeq \in \mathcal{B}/_\simeq$ is it-selves complement, $[b]_\simeq = [b]^c_\simeq$, this means that there exists a bijection $\pi \in \mathfrak{Bij}(b, b)$ such that $[i]^c_\simeq = [\pi(i)]_\simeq$. This means that the block defined by $b^* := \{i, \pi(i)\}$ is complete and $b^* \subseteq b$. Theorem 3.1.12 gives $[b^*]_\simeq \leq [b]_\simeq$, and therefore $[b^*]_\simeq = [b]_\simeq$ if $[b]_\simeq \in \mathcal{B}^U/_\simeq$. □

A similar condition holds for the completeness of a block

Theorem A.0.13. *A block type $[b]_{\simeq}$ is complete if and only if $[b]^c_{\simeq}$ is complete.*

Proof. Let $[b]_{\simeq} \in \mathcal{B}^C$ be a complete block and b^* a representative of $[b]^c_{\simeq}$. Then a bijection $\pi \in \mathfrak{Bij}(b, b^*)$ can be found which fulfils $P_{i,d} = 1 - P_{\pi(i),d}$ and it holds

$$H_{b,d} = \sum_{i \in b} P_{i,d} = \sum_{i \in b} (1 - P_{\pi(i),d}) = |b| - \sum_{i \in b^*} P_{i,d} = |b| - H_{b^*,d}.$$

By Definition 3.1.3 this leads to the claimed equivalence. □

The proof above leads to the following consequence

Corollary A.0.14. *If $[b]_{\simeq}$ is a complete block type and b^* a representative of $[b]^c_{\simeq}$, then the height of the complement block type fulfils the condition*

$$H_{b^*} = |b| - H_b.$$

The next theorem shows the relation between complete blocks and indivisible complete blocks.

Theorem A.0.15. *Every complete block can be written as a partition of indivisible complete blocks.*

Proof. The proof is a simple induction on the height h of a block, starting with the fact that all complete block of height 1 are indivisible.
Assuming a complete block b of height $h + 1$, it is either indivisible or has a sub block with height lower equal h. In the first case, we found a partition consisting of just this block. In the second case, we found a sub block b' which can be partitioned in indivisible blocks due to the inductive assumption. If we consider the customers of $b \backslash b'$, we see that they also form a complete block b'' with height lower equal h. The unification of the two partitions of block b' and b'' is a partition of b in indivisible blocks. □

The partition in indivisible blocks introduced in Theorem A.0.15 is not unique as the following simple example shows.

Example 3. *Assume a time horizon of $D = 2$, two customers 1 and 3 requiring service on day 1 and two customers 2 and 4 requiring service on day 2. Then the block $b = \{1, 2, 3, 4\}$ is complete and can be partitioned in two different ways, i.e.*

$$b = \{1, 2\} \dot{\cup} \{3, 4\} = \{1, 4\} \dot{\cup} \{3, 2\}$$

But it is possible formulating a result for the partition of a complete block regarding the block height.

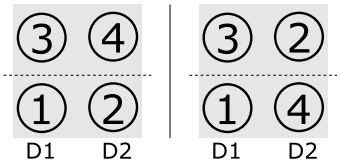


Figure A.2: Example for a not unique block partition

Theorem A.0.16. *Let $b \in \mathcal{B}^C$ be a complete block with partition $b = b_1 \dot{\cup} ... \dot{\cup} b_m$ and $b_k \in \mathcal{B}^U$ for all $k \in 1, ..., m$, then the following equation holds*

$$H_b = \sum_{k=1}^{m} H_{b_k}.$$

Proof. For all complete blocks holds $H_b = H_{b,d}$ for an arbitrary $d \in \mathcal{D}$, so the equation above follows from

$$H_{b,d} = \sum_{i \in b} P_{i,d} = \sum_{k=1}^{m} \sum_{i \in b_k} P_{i,d} = \sum_{k=1}^{m} H_{b_k,d}.$$

□

Corollary A.0.17. *Assuming a block $b \in \mathcal{B}^C$ to be divisible, then there is an indivisible sub block $b* \leq b$ with*

$$H_{b*} \leq \frac{H_b}{2}.$$

Proof. Similar to the notation of Theorem A.0.16 let $b = b_1 \dot{\cup} ... \dot{\cup} b_m$ and $b_k \in \mathcal{B}^U$. Assuming the contraposition that for all b_k holds $H_{b_k} > \frac{H_b}{2}$ results in

$$H_b = \sum_{k=1}^{m} H_{b_k} > \sum_{k=1}^{m} \frac{H_b}{2} \geq H_b.$$

The right inequality follows from the divisibility of b, giving $m \geq 2$, and therefore, it follows the contradiction. □

Appendix B

Block computation from the locational point of view

As mentioned in Section 3.1.2, a grouping $\mathcal{B}' \subseteq \mathcal{B}$ fulfilling the conditions

$$\bigcup_{b \in \mathcal{B}'} b = \mathcal{N} \tag{B.1}$$

$$b_1 \cap b_2 = \emptyset, \ \forall b_1, b_2 \in \mathcal{B}' \tag{B.2}$$

can be also chosen optimal from the locational point of view. Using the notation of MIP 6 a set $\mathcal{B}'$ should be found with minimal $\sum_{b \in \mathcal{B}'} \sum_{i \in \mathcal{N}} (g^{\text{lat}}_{b,i} + g^{\text{lon}}_{b,i})$ with $g^{\text{lat}}_{b,i} := |m^{\text{lat}}_b - G^{\text{lat}}_i|$ if $i \in b$ and equal 0 otherwise. The variables $g^{\text{lon}}_{b,i}$ are analogously defined. The value of these variables are known a priori, if and only if the customers assigned to block b are known and therefore, also the coordinates of the blocks artificial centre or geometric median. Then it is possible defining the exact costs of a block b by

$$T_b := \sum_{i \in b} (|G^{\text{lat}}_i - m^{\text{lat}}_b| + |G^{\text{lon}}_i - m^{\text{lon}}_b|).$$

Simply searching for a set $\mathcal{B}' \subseteq \mathcal{B}$ with minimal $\sum_{b \in \mathcal{B}'} T_b$ and fulfilling (B.1) and (B.2) would result in a set of blocks, each consisting of just a single customer. Conversely, assuming $\mathcal{B}' \subseteq \mathcal{B}^U$ would mean that one of the conditions, (B.1) and (B.2), would not hold any more, but it is possible computing a cover $\mathcal{B}'$ of $\mathcal{N}$ with indivisible, complete blocks by solving MIP 11. In a second step, indivisible, complete blocks of $\mathcal{B}'$ are replaced by incomplete blocks such that condition (B.2) is again fulfilled and the number of indivisible, complete blocks, i.e. $\mathcal{B}' \cap \mathcal{B}^U$, is maximal.

Mixed Integer Program 11. *minimizing*

$$\sum_{b \in \mathcal{B}^U} T_b u_b$$

subject to

$$\sum_{b \in \mathcal{B}^U} Z_{i,b} u_b \geq 1, \ \forall i \in \mathcal{N}$$

It easily can be seen that MIP 11 corresponds to the Linear Program 2 (also see [9]), if the terms of blocks are replaced by routes. Analogously, the binary variable u_b is equal 1 if block b is used, i.e. $b \in \mathcal{B}'$, and 0 otherwise and the constants $Z_{i,b}$ describe if $i \in b$ or not. It immediately suggest itself using a variant of the branch and price algorithm described in detail by Feillet [9].
For a subset $\mathcal{B}_0^U \subseteq \mathcal{B}^U$ the restricted Master Problem and its Dual Program is defined as follows.

Linear Program 12 (restricted Master Program)**.** *minimizing*

$$\sum_{b \in \mathcal{B}_0^U} T_b u_b$$

subject to

$$\sum_{b \in \mathcal{B}_0^U} Z_{i,b} u_b \geq 1, \ \forall i \in \mathcal{N}$$

$$u_b \geq 0, \ \forall b \in \mathcal{B}_0^U$$

Linear Program 13 (Dual Program)**.** *maximizing*

$$\sum_{b \in \mathcal{N}} \tilde{u}_i$$

subject to

$$\sum_{i \in \mathcal{N}} Z_{i,b} \tilde{u}_i \leq T_b, \ \forall b \in \mathcal{B}_0^U$$

$$\tilde{u}_i \geq 0, \ \forall i \in \mathcal{N}$$

The part of column generation focuses on the search of blocks with negative reduced costs. The negative reduced cost condition for a block b is given by

$$T_b - \sum_{i \in \mathcal{N}} Z_{i,b} \tilde{u}_i < 0.$$

If such a block is found, it is added to the set $\mathcal{B}_0^U$. Important to mention is that it can not happen that a block is added twice because every block in $\mathcal{B}_0^U$ does not fulfil the reduced cost condition (see [9][p. 413]).
Given a fixed block type $[b^*]_\simeq \in \mathcal{B}^U/_\simeq$ with representative b^* and a set of customer types $\{[i_1]_\simeq, ..., [i_m]_\simeq\}$, fulfilling $\exists i \in [i_k]_\simeq \ : \ i \in b^*$ for every $k \in \{1, ..., m\}$, then the following LP computes the block of type $[b^*]_\simeq$ with minimal reduced costs. Using the notation of MIP 6 and defining $\mathcal{N}_{[b^*]_\simeq} := \bigcup_{b \in [b^*]_\simeq} b$ the MIP is described as follows.

Mixed Integer Program 14 (Reduced Cost MIP)**.** *minimizing*

$$\sum_{i \in \mathcal{N}_{[b^*]_\simeq}} (g^{lat}_{b^*,i} + g^{lon}_{b^*,i}) - \sum_{i \in \mathcal{N}_{[b^*]_\simeq}} x_{b^*,i} \tilde{u}_i$$

subject to

$$g^{lat}_{b^*,i} \geq G^{lat}_i - m^{lat}_{b^*} - M(1 - x_{b^*,i}), \qquad \forall i \in \mathcal{N}_{[b^*]_\simeq} \tag{B.1}$$

$$g^{lat}_{b^*,i} \geq m^{lat}_{b^*} - G^{lat}_i - M(1 - x_{b^*,i}), \qquad \forall i \in \mathcal{N}_{[b^*]_\simeq} \tag{B.2}$$

$$g^{lon}_{b^*,i} \geq G^{lon}_i - m^{lon}_{b^*} - M(1 - x_{b^*,i}), \qquad \forall i \in \mathcal{N}_{[b^*]_\simeq} \tag{B.3}$$

$$g^{lon}_{b^*,i} \geq m^{lon}_{b^*} - G^{lon}_i - M(1 - x_{b^*,i}), \qquad \forall i \in \mathcal{N}_{[b^*]_\simeq} \tag{B.4}$$

$$N_{b^*,i_k} = \sum_{i \in [i_k]_\simeq} x_{b^*,i}, \qquad \forall k \in \{1, ..., m\} \tag{B.5}$$

$$g^{lat}_{b^*,i} \geq 0, \qquad \forall i \in \mathcal{N}_{[b^*]_\simeq} \tag{B.6}$$

$$g^{lon}_{b^*,i} \geq 0, \qquad \forall i \in \mathcal{N}_{[b^*]_\simeq} \tag{B.7}$$

$$m^{lat}_{b^*} \geq 0 \tag{B.8}$$

$$m^{lon}_{b^*} \geq 0 \tag{B.9}$$

$$x_{b^*,i} \in \{0, 1\}, \qquad \forall i \in \mathcal{N}_{[b^*]_\simeq} \tag{B.10}$$

If for every $[b^*]_\simeq \in \mathcal{B}^U/_\simeq$ the objective value of the LP above is greater equal 0, this means that no block with negative reduced costs exists or equivalently the last solution set $\mathcal{B}' \subseteq \mathcal{B}^U_0$ found by solving the restricted master problem is optimal (see [9]).

Appendix C

Minimality of the median

Theorem C.0.18. *Given an ordered, finite set of data points $\{G_1, ..., G_n\}$, then the median x_{med} of this set is the point with the minimal distance to all data points, i.e.*

$$x_{med} = \arg \min_{x \in [G_1, G_n]} \sum_{i=1}^{n} |x - G_i|$$

Proof. The idea for the following proof comes from [22]. Let $\mathcal{L}$, $\mathcal{M}$ and $\mathcal{U}$ be defined as follows

$$\begin{aligned} \mathcal{L} &:= \{i \in \{1, ..., n\} : G_i < x_{med}\} \\ \mathcal{M} &:= \{i \in \{1, ..., n\} : G_i = x_{med}\} \\ \mathcal{U} &:= \{i \in \{1, ..., n\} : G_i > x_{med}\}, \end{aligned}$$

then the sum of distances can be written as

$$\Delta(x_{med}) := \sum_{i=1}^{n} |x_{med} - G_i| = \sum_{i \in \mathcal{L}} (x_{med} - G_i) + \sum_{i \in \mathcal{U}} (G_i - x_{med}).$$

Subsequently, define $\tilde{x} := x_{med} + \epsilon$ with $\epsilon > 0$, then $\mathcal{U}$ is decomposed into $\mathcal{U}_< := \{i \in \mathcal{U} : G_i < x_{med} + \epsilon\}$ and $\mathcal{U}_\geq := \{i \in \mathcal{U} : G_i \geq x_{med} + \epsilon\}$ and the sum of distances to $\tilde{x}$ is equal to

$$\begin{aligned} \Delta(\tilde{x}) = \sum_{i \in \mathcal{L}} (x_{med} + \epsilon - G_i) + \sum_{i \in \mathcal{M}} \epsilon + \sum_{i \in \mathcal{U}_<} (x_{med} + \epsilon - G_i) + \sum_{i \in \mathcal{U}_\geq} (G_i - x_{med} - \epsilon) = \\ = \Delta(x_{med}) - 2 \sum_{i \in \mathcal{U}_<} (G_i - x_{med}) + \epsilon(|\mathcal{L}| + |\mathcal{M}| + |\mathcal{U}_<| - |\mathcal{U}_\geq|) \end{aligned}$$

For every $i \in \mathcal{U}_<$ holds that $G_i - x_{med} < \epsilon$ and therefore

$$\Delta(\tilde{x}) > \Delta(x_{med}) + \epsilon(|\mathcal{L}| + |\mathcal{M}| + |\mathcal{U}_<| - |\mathcal{U}_\geq| - 2|\mathcal{U}_<|) = \Delta(x_{med}) + \epsilon(|\mathcal{L}| + |\mathcal{M}| - |\mathcal{U}|)$$

The property of the median gives that $|\mathcal{L}| + |\mathcal{M}| - |\mathcal{U}| > 0$ resulting in $\Delta(x_{med} + \epsilon) > \Delta(x_{med})$ for positive ϵ.
The proof for $\epsilon < 0$ work analogously with the difference that $\mathcal{L}$ has to be decomposed. In summary, $\Delta(x_{med} + \epsilon) > \Delta(x_{med})$ holds for an arbitrary $\epsilon \in \mathbb{R}$, which proofs the minimality of the median. □

Bibliography

[1] Z. Borčinova. Two models of the capacitated vehicle routing problem. *Croatian Operational Research Review*, 8(2):463–469, 2017.

[2] K. Braekers, K. Ramaekers, and I. Van Nieuwenhuyse. The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, 99:300 – 313, 2016. ISSN 0360-8352. doi: https://doi.org/10.1016/j.cie.2015.12.007. URL http://www.sciencedirect.com/science/article/pii/S0360835215004775.

[3] P. Campelo, F. Neves-Moreira, P. Amorim, and B. Almada-Lobo. Consistent vehicle routing problem with service level agreements: A case study in the pharmaceutical distribution sector. *European Journal of Operational Research*, 273(1):131 – 145, 2019. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2018.07.030. URL http://www.sciencedirect.com/science/article/pii/S0377221718306350.

[4] N. Christofides and J. Beasley. The period routing problem. *Networks*, 14(2):237–256, 1 1984. ISSN 1097-0037. doi: 10.1002/net.3230140205. URL http:https://doi.org/10.1002/net.3230140205.

[5] G. Dantzig and J. Ramser. The truck dispatching problem. *Management Science*, 6 (1):80–91, 1959. doi: 10.1287/mnsc.6.1.80. URL https://doi.org/10.1287/mnsc.6.1.80.

[6] G. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, 2(4): 393–410, 1954. ISSN 00963984. URL http://www.jstor.org/stable/166695.

[7] G. Desaulniers, O. Madsen, and S. Røpke. *The Vehicle Routing Problem with Time Windows*, pages 119–160. Society for Industrial and Applied Mathematics, 2 edition, 2014. ISBN 978-1-611973-58-7.

[8] M. Desrochers. *An algorithm for the shortest path problem with resource constraints*. Université de Montréal, Centre de recherche sur les transports, 1986.

[9] D. Feillet. A tutorial on column generation and branch-and-price for vehicle routing problems. *4OR*, 8(4):407–424, Dec 2010. ISSN 1614-2411. doi: 10.1007/s10288-010-0130-z. URL https://doi.org/10.1007/s10288-010-0130-z.

[10] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004. doi: 10.1002/net.20033. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/net.20033.

[11] D. Feillet, M. Gendreau, and L. Rousseau. New refinements for the solution of vehicle routing problems with branch and price. *INFOR: Information Systems and Operational Research*, 45(4):239–256, 2007. doi: 10.3138/infor.45.4.239. URL https://doi.org/10.3138/infor.45.4.239.

[12] D. Feillet, T. Garaix, F. Léhuéde, O. Péton, and D. Quadri. A new consistent vehicle routing problem for the transportation of people with disabilities. *Networks*, 63(3): 211–224, 2014. ISSN 1097-0037. doi: 10.1002/net.21538. URL http://dx.doi.org/10.1002/net.21538.

[13] B. Gittenberger. Skriptum zur vorlesung diskrete methoden, April 2010.

[14] C. Groër, B. Golden, and E. Wasil. The consistent vehicle routing problem. *Manufacturing & Service Operations Management*, 11(4):630–643, October 2009. ISSN 1526-5498. doi: 10.1287/msom.1080.0243. URL http://dx.doi.org/10.1287/msom.1080.0243.

[15] G. Hiermann, J. Puchinger, S. Røpke, and R. Hartl. The electric fleet size and mix vehicle routing problem with time windows and recharging stations. *European Journal of Operational Research*, 252(3):995 – 1018, 2016. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2016.01.038. URL http://www.sciencedirect.com/science/article/pii/S0377221716000837.

[16] R. Keith. The marketing revolution. *Journal of Marketing*, 24(3):35–38, 1960. ISSN 00222429. URL http://www.jstor.org/stable/1248704.

[17] A. Kovacs, B. Golden, R. Hartl, and S. Parragh. Vehicle routing problems in which consistency considerations are important: A survey. *Networks*, 64(3):192–213, 2014. ISSN 1097-0037. doi: 10.1002/net.21565. URL http://dx.doi.org/10.1002/net.21565.

[18] A. Kovacs, S. Parragh, and R. Hartl. A template-based adaptive large neighborhood search for the consistent vehicle routing problem. *Networks*, 63(1):60–81, 2014. ISSN 1097-0037. doi: 10.1002/net.21522. URL http://dx.doi.org/10.1002/net.21522.

[19] A. Kovacs, B. Golden, R. Hartl, and S. Parragh. The generalized consistent vehicle routing problem. *Transportation Science*, 49(4):796–816, 2015. doi: 10.1287/trsc.2014.0529. URL http://dx.doi.org/10.1287/trsc.2014.0529.

[20] A. Kovacs, S. Parragh, and R. Hartl. The multi-objective generalized consistent vehicle routing problem. *European Journal of Operational Research*, 247(2):441 –

458, 2015. ISSN 0377-2217. doi: http://dx.doi.org/10.1016/j.ejor.2015.06.030. URL http://www.sciencedirect.com/science/article/pii/S0377221715005445.

[21] Z. Luo, H. Qin, C. Che, and A. Lim. On service consistency in multi-period vehicle routing. *European Journal of Operational Research*, 243(3):731 – 744, 2015. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2014.12.019. URL http://www.sciencedirect.com/science/article/pii/S0377221714010200.

[22] Matroids Matheplanet. Beweis fuer die minimaleigenschaft des medians. https://matheplanet.com/default3.html?call=viewtopic.php?topic=197037&ref=https%3A%2F%2Fwww.google.com%2F, 2014.

[23] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *Proceedings of the 4th International Conference on Principles and Practice of Constraint Programming*, CP '98, pages 417–431, London, UK, UK, 1998. Springer-Verlag. ISBN 3-540-65224-8. URL http://dl.acm.org/citation.cfm?id=647485.726320.

[24] A. Subramanyam and C. Gounaris. A branch-and-cut framework for the consistent traveling salesman problem. *European Journal of Operational Research*, 248(2):384 – 395, 2016. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2015.07.030. URL http://www.sciencedirect.com/science/article/pii/S0377221715006633.

[25] P. Toth and D. Vigo. *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, 2002. ISBN 978-1-611973-58-7. doi: 10.1137/1.9780898718515.ch1.

[26] H. Zhong, R. Hall, and M. Dessouky. Territory planning and vehicle dispatching with driver learning. *Transportation Science*, 41(1):74–89, 2007. doi: 10.1287/trsc.1060.0167. URL https://doi.org/10.1287/trsc.1060.0167.