universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Exact solution approaches for the collaborative vehicle routing problem“

verfasst von / submitted by

Philipp E.H. Salzmann, BSc.

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

Master of Science (MSc.)

Wien 2017 / Vienna 2017

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 066 915
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Betriebswirtschaftslehre
Betreut von / Supervisor:	O.Univ.Prof. Dipl.-Ing. Dr. Richard F. Hartl
Mitbetreut von / Co-Supervisor:	Mag. Dr. Margaretha Gansterer

Acknowledgement I would like to express my gratitude and appreciation to Professor Hartl, my research supervisor for the thesis at hand, and Professor Dörner, my supervisor in the years to come. In addition I would like to thank Dr. Gansterer, who took a key role in the completion of this thesis, for her patient guidance while keeping the schedule somehow on track.

I would like to extend my thanks to the rest of the departments for Production and Logistics and for Production and Logistics with international Focus, they offered guidance, encouragement, and useful critics. Especially Biljana Roljic for the time she spent on proofreading.

Finally, I wish to thank my family and especially my parents, who always encouraged me to take my own decisions in live and supported me on the path to accomplish those. I am thankful for the opportunities my parents provided me.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

The thesis at hand is focusing on modelling and solving a problem from the sphere of last mile delivery. One of the most common problems in city logistics is the delivery of parcels and similar logistic units. Usually multiple service providers will be present in the same area and act as opponents in the same market. Those markets for the most part are highly competitive in terms of prices and customer service. In such competitive scenarios carrier collaboration might be an option. Each of the carriers could improve its productivity, cost structure, and customer service. Those effects would come at low or no costs since the required infrastructures are already present.

However, some carriers still have reservations towards collaborative vehicle routing. The main reason for those are trust issues. Those manifest in the unwillingness of the carriers to exchange or reveal information about their customers and contracts. By doing so they fear to lose their competitive advantage. Regardless of the overt advantages, resentments often triumph over the pledges of collaboration.

The following chapters will deal with one of the most common transportation problems in real world application: the transportation of packages or letters by different service providers from one customer to another. Such a service provider could be any carrier transporting goods directly from an origin to a destination without relocating the packages between different vehicles or carriers. One example could be multiple bike messengers (with an infinite capacity) or a group of self-employed truck drivers. In the recent past providers were forced to increase the service level and feasibility as well

as to reduce the lead times and costs. New solution approaches tackling those developments are necessary to maintain competitiveness among carriers. Dealing with those challenges by acquiring new and more efficient technologies (e.g. oversize trucks or sails on container ships) will require investments and time for implementation. Collaborating carriers could increase their operational flexibility and competitiveness while decreasing their costs. The structures required to exchange transportation requests are often already existing and could be used on a regular basis as well.

The research done here is motivated by recent publications considering horizontal collaboration in the literature for vehicle routing. Horizontal collaboration describes partnerships among companies that are located on the same level of the supply chain. One example would be the exchange of transportation requests between different courier services in the same city or region. According to Gansterer and Hartl [2017] collaborative vehicle routing is an active research area with high potentials in terms of cost and emission savings.

The transportation of small shipments (e.g. packages) is particularly interesting for horizontal collaboration. Since packages of different requests, carriers, or customers can be transported on the same vehicle at the same time, the degree of collaboration and freedom is much higher than for example in full truckload transportation planning according to Archetti et al. [2014] and Gansterer and Hartl [2017].

An important question in this context is how to motivate providers to cooperate and how to share the gain. This is why the field of collaboration gets increasingly important in transportation science. And in the last years a few

papers already covered this topic and tried to developed strategies for efficient transportation through carrier collaboration. The most common problem in this field is how to (re-)assign transportation contracts in the network and how to deal with information inside such a network. Since the carriers are competitors, they hesitate to share information about their cost structures and customer base. A few papers assume a neutral agent reassigning the jobs inside the system an setting the compensations and therefore the profit sharing. Those scenarios are referred to as *centralized collaborative planning approaches*. For this purpose, a neutral instance would get all the information associated with the requests from every carrier, while guaranteeing to the carriers that they will not use those to their disadvantage. Dai and Chen [2012a] suggest an online platform as such a neutral instance. A neutral centralized instance would solve a giant vehicle routing problem (VRP) with pickup and delivery (VRPPD). The carriers would get the information on which contracts would be performed by which of the carriers.

For a decentralized approach the carriers have to reveal only a small proportion of the information they would have to reveal in a centralized approach. This could increase the acceptance by the practitioners. For both approaches centralized as well as decentralized the most important questions are how to divide the gain from collaboration and in the first place how to (re-)assign the requests . For horizontal collaboration Berger and Bierwirth [2010] and Gansterer and Hartl [2016] proposed and tested multiple auctions for the collaborative carrier routing problem. Those auctions like other decentralized solution approaches will most probably create constellations that are worse than a solution based on a centralised collaborative planning

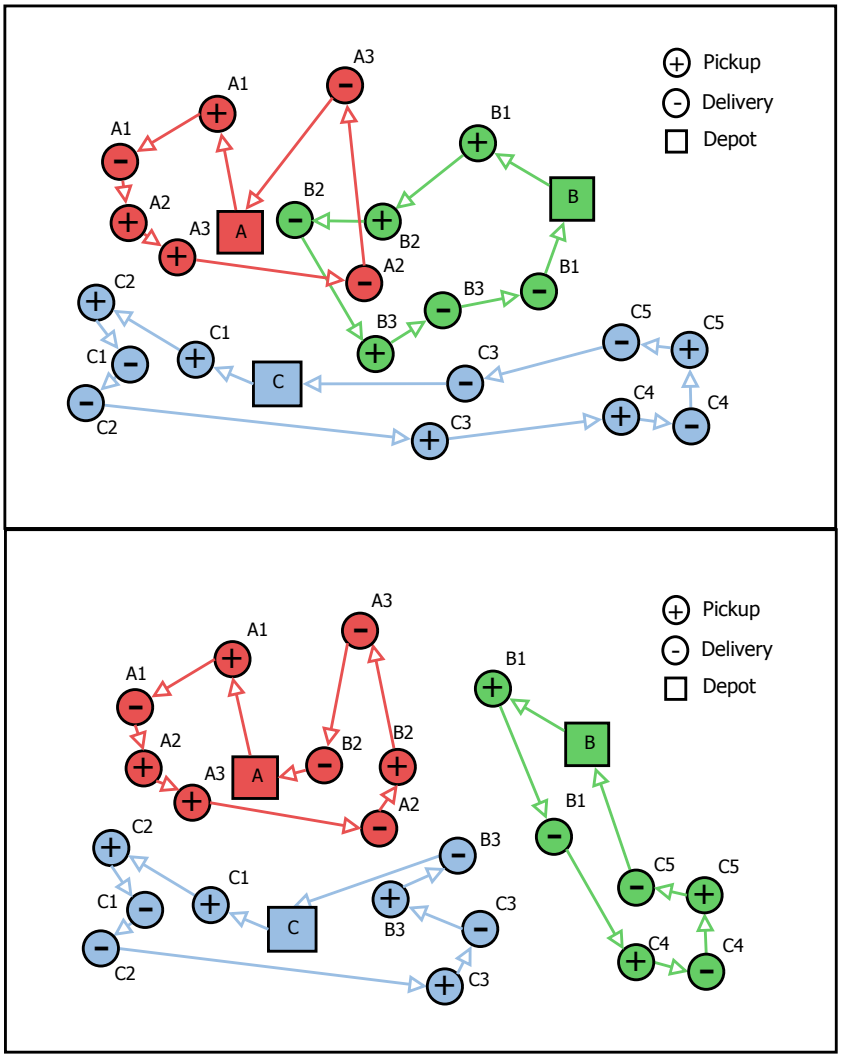
approach. Even though centralised collaborative planning approaches will hardly be used in practice due to trust issues in competitive environments, solutions to the centralized problem are still important as a benchmark for decentralized mechanisms.

In a collaborative carrier routing problem like described in the paper of Berger and Bierwirth [2010] or Gansterer and Hartl [2016] a classical routing problem with multiple carriers gets solved. Each of the carriers has its own depot and vehicle. Since there is no time or capacity constraint, there is no scenario where more than one vehicle per depot would be beneficial. Every carrier has multiple contracts with customers which consist of a pickup location and a delivery location. Neither a capacity restriction nor a quantity is given for those contracts. There is a profit for every contract depending on the distance between pickup and delivery location.

Figure 1 shows a scenario with and without collaboration for a routing problem with three carriers. The proposed solution is based on equal workloads. In the proposed solution every carrier would clearly benefit from the exchanges. The scenario shown is similar to those solved later on in the computational results section.

When transferring this problem to a central planning approach one will end up with a routing problem with as many depots as carriers and one pool of requests. In contrast to the more common pickup and delivery problems in research, there are neither capacity nor time constraints. This problem will be mathematically modelled and solved with multiple exact solution approaches in the thesis at hand.

Figure 1: This figure compares the result of an non collaborative scenario with the situation with collaboration. (Gansterer and Hartl [2017])



An important issue for the collaboration of the different carriers is the workload distribution. In real world application, carriers will be interested in acquiring a certain percentage of requests. One reason is their interest in utilizing their infrastructure, the other to feignedly improve their profits or income by getting compensations. Some researchers argue that the carriers are more willing to accept solutions from a collaborative scenario, when they are able to use their own infrastructure to a certain point or percentage. The previously proposed collaborative carrier routing problems are not taking such considerations into account.

The thesis at hand contributes to research by providing the following insights:

- An extensive comparison of exact approaches for a special case of the Pickup and Delivery Problem (PDP) that reveals the strengths and weaknesses of those methods.
- Presenting a way to deal with unevenly distributed workloads as well as investigating the associated effects. The trade-off between solution quality and level of workload distribution is analysed.
- Offering new benchmarks for heuristic and exact approaches for the centralized and the decentralized planning approaches.

The exact approaches tested further on are multiple Branch and Cut variations, two different Benders Decomposition approaches, and a Column Generation approach. Those approaches are founded on the work done for similar problems in the literature.

The thesis is organized as follows. Chapter 2 is describing the problem that was solved further on. A literature review focusing on collaborative routing problems and problems from the sphere of pickup and delivery problems is presented in Chapter 3. Mathematical models for the proposed problems are presented in Chapter 4. Chapter 5 describes the exact methods that were applied to the problem. Computational analysis concerning the run time and routing costs are presented in Chapter 6. The last chapter sums up the findings of the thesis and offers recommendations for future research.

2 Problem description

The problem studied here was originally proposed by Berger and Bierwirth [2010] in the context of collaborative carrier routing problems. In collaborative carrier routing problems multiple carriers have to fulfil a certain number of jobs that are originally assigned to one of them. Each carrier has a fleet of a certain size and their own depot location. In the problem setting used by Berger and Bierwirth [2010] and Gansterer and Hartl [2016] the number of carriers is three and every carrier has one depot and one vehicle. In addition there is a certain number of jobs, each of them originally assigned to one of the carriers. The jobs consist of a pickup and a delivery location. The profit that can be obtained with each of the jobs depends on the distance between the pickup and the delivery location. Since all jobs have to be serviced and the profit is independent from the assignment, the problem can be solved as a minimization problem. Therefore, the objective is to minimize the routing costs instead of maximizing the profit. In general the collaborative carrier routing problem could be combined with other constraints like time windows or some kind of capacity constraint. In the collaborative carrier routing problem as proposed by Berger and Bierwirth [2010] those or similar constraints are not present.

Trying to solve the problem proposed by Berger and Bierwirth [2010] one may end up dealing with a routing problem that on the one side includes multiple depots and paired pickup and delivery jobs but on the other side neglects any further constraints. Based on that knowledge one may already

see that the problem is pretty unrestricted and only sub-tour elimination and precedence constraints are restricting the problem at hand.

But one major drawback from real world applications is not considered in the models proposed. A relevant proportion of the carriers will insist on utilising their own resources and will ask for a certain percentage of the overall or their own workload. Those requests may decrease the potential gain from collaboration. The model solved here was amended with constraints that take those restrictions into account.

Summing up, the problem at hand is a Multi Depot Travelling Salesman (or Vehicle Routing Problem) with Pickup and Delivery (MDTSPPD). Each depot has one vehicle assigned. Because of the missing capacity and time window restrictions, there is no additional gain from considering multiple vehicles per depot. The requests that have to be serviced consist of a pickup location as well as a delivery location. We deal with a so called paired pickup and delivery problem. The only additional constraint is the so called minimum customer or workload constraint that requires every carrier to fulfil a certain number of requests.

3 Literature review

The following literature review will primarily focus on two different areas of research. One offering additional information about the collaborative carrier routing problem, describing different approaches presented in recent research and the possible impact to real world applications. The other trying to classify the problem inside the general research for vehicle routing problems and giving a brief overview over the field of PDP as well as the solution methods deployed on problems similar to the one presented here.

Collaboration in Transportation The problem at hand has its origin in the literature and research of collaboration in transportation. Krajewska and Kopfer [2006] and Cruijssen et al. [2007] first studied the potential benefits form collaboration between different carriers. Collaboration in this context means to exchange requests between different carriers. In general one may distinguish between two scenarios of collaboration: Less than truckload and full truckload.

In full truckload collaboration empty rides offer an opportunity to reduce the costs of the participating carriers. The literature proposes joint route planning and route merging as collaborative tools. Joint route planning was investigated by Dai and Chen [2012a], Dai and Chen [2012b], Buijs et al. [2016], and Liu et al. [2010]. Merging preplanned routes as a collaboration approach was done by Adenso-Díaz et al. [2014], Ergun et al. [2007], and Kuyzu [2017]. Those papers focus on merging routes, that each carrier on its own would consider a full load. A variety based on the same set of assumptions can be found with Liu et al. [2010] where capacitated vehicles

and multiple depots were also considered for a routing problem in carrier collaboration with full truckloads.

The literature for less than truckload is mostly dealing with collaboration in courier services. A selection of the research done in this field can be found in Berger and Bierwirth [2010], Buijs et al. [2016], Gansterer and Hartl [2016], and Nadarajah and Bookbinder [2013]. For a real world setting you may consult Lin [2008].

The research described so far mainly focuses on the profit or cost aspects while other papers like Montoya-Torres et al. [2016], Pérez-Bernabeu et al. [2015], and Sanchez et al. [2016] also consider pollution, noise and other negative or harmful side effects. But keeping in mind, that costs and profit as well as costs and distance have a strong connection and that those side effects may be represented as cost matrices in a reasonable way. This makes this research a special case of the more general collaborative scenarios.

The problems solved in literature are usually decomposed in different stages to tackle the real world problem size. For less than truckload Dai and Chen [2012b] present such two stage approaches. Dai and Chen [2012b] combine a mixed integer problem with a lane covering problem and than construct feasible routes from this combination.

By relaxing the usual assumption that the request may not change the vehicle between origin and destination Buijs et al. [2016] present a real world application for the Dutch logistic provider *Fritom* for in house collaboration of different business units. Multiple decomposition strategies were considered to tackle the real world instances.

Wang et al. [2014] combined the horizontal collaboration with vertical collaboration via subcontracting and request exchange. Comprehensive literature reviews can be found with Verdonck et al. [2013] and Gansterer and Hartl [2017].

Another field of interest from research that has to be considered is the class of pickup and delivery problems The collaborative carrier problem with less than truckload will share most of its features with this problem class.

Pickup and Delivery Problems The PDP in its most general form (according to literature) can be observed in Savelsbergh and Sol [1995]. In the proposed paper the authors try to give a generalised model that incorporates the characteristics of all the problems in the sphere of routing with pickup and delivery. The proposed General Pickup and Delivery Problem has a start depot and an end depot as well as a given capacity per vehicle. According to Savelsbergh and Sol [1995], the following problems can be seen as special cases of the generalized version:

- The Pickup and Delivery Problem (PDP) where a certain number of homogeneous vehicle fulfil a certain number of requests. Each of the requests consists of an origin, a destination, and a quantity.

- The Dial-a-Ride (DARP) is very similar, though it deals with persons instead of goods. For more information consult Cordeau and Laporte [2007].

- The Vehicle Routing Problem (VRP) a version of the Travelling Salesman Problem (TSP) with the opportunity that multiple vehicles can

be used to fulfil the requests. The special case in the VRP compared to the PDP is that the pickup or the delivery locations of all requests are at the depot. In a VRP a vehicle will just bring goods from the different destination to the depot or from a depot to certain destinations but never transport any amount in between two destinations. For additional and more specific information about this standard problem of operations research and logistics consult Laporte [1992], Golden et al. [2008], and Toth [2014].

In addition to the generalisation of the problem Savelsbergh and Sol [1995] propose the following problem characteristics to differentiate the variations in the sphere of PDP:

- Planning horizon (static vs. dynamic request set)
- Different types of time windows and time constraints
- Different objectives: duration, completion time, travel time, route length, client inconvenience, number of vehicles, and profit.

In the review which is part of the Savelsbergh and Sol [1995] paper, exact and heuristic approaches for the static and the dynamic problem with and without time windows were proposed. Exact approaches were cited and proposed only for the dial-a-ride problem with a single vehicle or for problems with time windows. For problems similar to the one at hand no exact approach is cited.

Another comprehensive review for the Pickup and Delivery Problem is Parragh et al. [2008a]. According to Parragh et al. [2008a] you can classify the problems derived from the General Pickup and Delivery Problem into

two major groups: VRP with backhaul (VRPB) and VRP with pickup and delivery (VRPPD). The problems of the VRPB kind deal with transportation requests only from or to a depot, or both at once but there are no requests that require a transportation from one customer to another. For more information about VRPB consult Parragh et al. [2008b]. The problems of the VRPPD kind can be structured into paired and unpaired requests according to Parragh et al. [2008b]. Unpaired requests allow pickup and deliveries at every customer and there are homogeneous goods, which means you could serve a demand at a certain customer with the supply picked up at another customer. A recent example for such a feature would be Turan et al. [2017]. Here for paired requests a certain object (commodity or person) has to be transported from a certain origin to a certain destination. The problem of the paired kind can be furthermore divided to Dial-a-Ride Problems (DARP) and Pickup and Delivery Problems (PDP). Parragh et al. [2008b] proposed multiple different methods of heuristic and exact nature. For the heuristic ones consult the review. Other reviews for the PDP can be found in Berbeglia et al. [2007] and Berbeglia et al. [2010].

The literature offers multiple exact approaches to the PDP. Kalantari et al. [1985] developed a Branch-and-Bound approach for the classical problem definition. For the PDP with time windows Dumas et al. [1991] presented a Column Generation approach, Ropke et al. [2007] a Branch-and-Cut approach, and as well as Ropke and Cordeau [2009], Baldacci et al. [2011] and Baldacci et al. [2010] present Branch-and-Cut-and-Price approaches. The PDP with shuttle routes got solved by Masson et al. [2014] and the PDP with multiple stacks by Cherkesly et al. [2015] both using Branch-and-Cut-

and-Price algorithms as well. Cordeau et al. [2010] solved a PDP with time windows, capacity constraints, and loading constraints with a Branch-and-Cut approach. Xue et al. [2016] use a Branch-and-Cut and a Branch-and-Price to a similar PDP with loading costs instead of loading constraints. With Lu and Dessouky [2004b] a PDP with multiple depots, time windows, and capacity constraints got solved with a Branch-and-Cut algorithm.

Other research that considers multiple depots but is not taking pickup and delivery precedence requirements in consideration, can be found with e.g. Dondo and Cerdá [2007] and Currie and Salhi [2003]. Salhi and Nagy [1999] added backhauls to the problem and Nagy and Salhi [2005] simultaneous pickup and delivery. PDP with heterogeneous fleets were tackled by Irnich [2000]. An heterogeneous fleet with soft time windows can be found with Bettinelli et al. [2014] and an heterogeneous fleet in a Dial-and-Ride setting by Detti et al. [2017]. In addition to those classic multi depot problems there is also a class of PDP with multiple regions. A survey and topology can be found with Dragomir et al. [2017].

To the best of the author's knowledge the next chapters will offer the first direct comparison of different exact solution methods for a problem of the proposed or a similar setting. Furthermore, the thesis at hand will be the first attempt to use tailored exact approaches to the MDTSPPD in general, and the first time that minimum workload restrictions are considered, taking real world application into account.

4 Problem formulations

In the following chapter two mathematical formulations will be presented. One of them is a vehicle flow formulation and the other a path based formulation. The vehicle flow formulation is used for the MIP solver, the Branch and Cut as well as for the Benders Decomposition approaches. The path based formulation is required for the Column Generation approach.

For the vehicle flow formulation two concepts were tested. One based on a Vehicle Routing Problem formulation and one based on a Travelling Salesman Problem formulation. The TSP based formulation is based on a Hamiltonian tour formulation and requires less decision variables and only a small proportion of the constraints required in the VRP based formulation. The TSP based formulation outperformed the VRP based formulation for all vehicle flow based methods. Berger and Bierwirth [2010] propose a very similar one that was also used in Gansterer and Hartl [2016].

The second formulation is required for the applied column generation approach. A column generation approach decomposes the problem in two parts. The first part is generating valid paths for the problem and the second is selecting the cost optimal set of valid paths. For this selection a path based formulation is proposed. In addition, the path based formulation generates dual costs. Those are required for the path generation. For more information about the algorithm you may consult Chapter 5. The formulation of the mathematical model is similar to routing problems, specifically the VRP.

In addition, a set of constraints is presented for the vehicle flow formulation, that could be used for multiple purposes. For example, they could be

used for workload considerations. For the path based formulation the sub problem will consider those constraints.

In the following two chapters those two mathematical problem formulations will be discussed. The vehicle flow formulation is a complete mathematical model while the proposed path based formulation will only consider the path selection. How to obtain the set of valid paths is presented in the method section. The criteria for a valid route can be determined based on the vehicle flow formulation as accustomed in the literature.

4.1 Vehicle flow formulation

The proposed vehicle flow formulation is based on the model proposed by Berger and Bierwirth [2010] and Lu and Dessouky [2004b]. It is a Travelling Salesman Problem formulation that incorporates multiple depots, pickup and delivery, and in addition workload restrictions. The model at hand is a combination of the Collaborative Carrier Routing Problem by Berger and Bierwirth [2010] and the Multi Vehicle Pickup and Delivery Problem as described by Lu and Dessouky [2004b]. The workload assumptions are new to the proposed problem class.

In contrast to Berger and Bierwirth [2010] the problem is formulated as a minimisation problem. This is possible because the profits used in the original formulation are not depending on the request assignment or the routing. The profits are constants and just depend on the the distance between pickup and delivery location.

In general, there are two options to model the problem at hand as previously proposed: VRP or TSP based. Using a classical Vehicle Routing Problem is the most common strategy to model a routing problem with multiple vehicles or depots. But since there are no vehicle capacity restriction in terms of distance, time, or load, there is no reason for an optimal solution to utilize more than one vehicle at each of the depots. This makes a TSP based formulation a reasonable option for the mathematical model. Using a TSP based formulation has major advantages concerning the number of decision variables as well as the number of constraints. A VRP based formulation will require decision variables three times the square of the number of customer plus one while the TSP based formulation will only require square of the number of customer plus the number of depots. The number of constraints will behave similar. Both will have an impact on the solution time of certain algorithms applied. A small disadvantage of the TSP based formulation arises from the pickup and delivery setting. The TSP based formulation will require precedence decision variables and constraints of set size square of the number of customers. A VRP based formulation could solve this feature with less effort in terms of constraints and variables, but would be more complex overall.

Comparing both formulations in terms of run times showed, that the TSP based outperforms the VRP based formulation. This observation holds for all methods that are based on the flow formulation, namely the MIP solver, all the proposed Branch and Cut, and the Benders Decomposition approaches.

The following paragraphs will discuss all the components required in the vehicle flow formulation.

Distance matrix An important part of the model is the modified distance matrix as suggested in Berger and Bierwirth [2010]. This is necessary to deal with the multiple depots. An additional advantage of the modified distance matrix is that the set of constraints can be downsized. Some of the constraints can be handled by so called big M values (high numbers that punish certain decisions) in the distance matrix. By doing so, some constraints of the original constraint set become redundant. The following distance matrix was applied in combination with the proposed mathematical formulation:

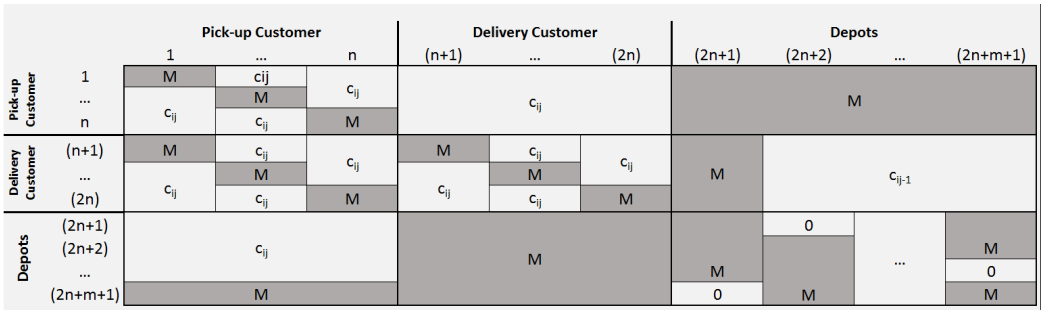


Figure 2: Modified distance matrix (Berger and Bierwirth [2010])

The distance matrix is of size square of two times the customers plus the number of depots plus one. The size results from the problem structure. On both axes of the distance matrix the first entries are the pickup locations followed by the delivery locations. When n is the number of contracts the pickup location x and delivery location (x + n) are member of the same paired request. The rest of the nodes on both axes are the depots. Since a Hamiltonian tour formulation is applied each of the depot nodes equals a start of a vehicle and an end of another vehicle. So the first depot node is the start of the first vehicle while the second depot is the start of the second vehicle

and the end of the first vehicle. Resulting from this definition there is one more depot node than actual depots or cars. The most of the connections are real world or Euclidean distances. But some of them represent a constraint or a logical connection. The scenarios that are punished with big M values are:

- the connection of a node with itself.
- all connections between a pickup location and a depot - since it is not allowed to enter a depot with a request
- all connections from a depot to any delivery node - since deliveries can only be performed when a pickup was performed
- all connections from a certain delivery to its pickup node - since those actions can be performed only in one direction
- almost all connections between depots are forbidden - since the only depot a vehicle can finish its tour at is by definition the direct successor

For the rest of the connections the distance between the two coordinates is used except for the following two exceptions: The costs between a node representing the start and the end of the same vehicle is zero since one is not forced to use multiple vehicles in the solution.

Mathematical formulation The mathematical model for the MDTSPPD is formulated based on the models proposed in Berger and Bierwirth [2010] and Lu and Dessouky [2004b] using the notation similar to the one from Gansterer and Hartl [2016]. The parameters and sets required are:

n ... number of customers

m ... number of depots

P ... set of pickup vertices, $P = \{1, \ldots, n\}$

D ... set od delivery vertices, $D = \{n+1, \ldots, 2n\}$

W ... set of depot vertices, $W = \{2n+1, \ldots, 2n+m+1\}$

N ... set of all vertices, $N = P \cup D \cup W = \{1, \ldots, 2n+m+1\}$

A ... set of all arcs, $A = N \times N$

c_{ij} ... cost for the arc from vertice i to vertice j

The decision variables used are:

x_{ij} ... decision variable indicating whether arc ij is used

b_{ij} ... decision variables indicating whether vertice i is visited before j

As previously discussed, the objective is to minimize the sum of the routing costs:

$$\min \sum_{(i,j) \in A} x_{ij} c_{ij} \tag{1}$$

Since all deliveries and pickups as well as all depot nodes have to be serviced exactly once the following constraints are necessary:

$$\sum_{i \in N} x_{ij} = 1 \qquad \forall j \in N \tag{2}$$

$$\sum_{j \in N} x_{ij} = 1 \qquad \forall i \in N \tag{3}$$

The next set of constraint is used to copy the values of the routing decision variables to the precedence decision variables:

$$b_{ki} \leq b_{kj} + (1 - x_{ij}) \tag{4}$$

$$\forall (i,j) \in A/\{2n+m+1, 2n+1\}, k \in N\backslash\{i\} \tag{5}$$

$$b_{kj} \leq b_{ki} + (1 - x_{ij})$$

$$\forall (i,j) \in A/\{2n+m+1, 2n+1\}, k \in N\backslash\{i\} \tag{6}$$

$$x_{ij} \leq b_{ij} \qquad \forall (i,j) \in A$$

The next set of constraints is necessary to align the pickup and the delivery nodes of the same customer as well as the depots based on the precedence decision variables. The constraints required are:

$$b_{ii} = 0 \qquad \forall i \in N \tag{7}$$

$$b_{n+i,i} = 0 \qquad \forall i \in P \tag{8}$$

$$b_{i,i+n} = 1 \qquad \forall i \in P \tag{9}$$

$$b_{ij} = b_{n+i,j} \qquad \forall i \in P, j \in W \tag{10}$$

$$b_{i,2n+1} = 0 \qquad \forall i \in N \tag{11}$$

$$b_{ij} = 1 \qquad \forall i,j \in W \mid i < j \tag{12}$$

$$b_{ji} = 0 \qquad \forall i,j \in W \mid i < j \tag{13}$$

$$b_{i,2n+m+1} = 1 \qquad \forall i \in N\backslash\{2n+m+1\} \tag{14}$$

$$x_{ij} = \{0,1\} \qquad \forall i,j \in N \tag{15}$$

$$b_{ij} = \{0,1\} \qquad \forall i,j \in N \tag{16}$$

Constraint (7), (11), (12), (13), and (14) are implicitly defined through the distance matrix and are not necessary for solving the mixed integer linear program. Constraint (10) is used to ensure that the pickup and the delivery of the same contract are scheduled for the same vehicle. While constraints (12) and (13) ensure the direction of the depots, constraint (14) ensures that the depot $(2n + m + 1)$ is the last node in the Hamiltonian tour. While it is necessary to ensure that the routing decision variable x_{ij} is binary, one may relax the definition of the precedence variable according to the proof that can be found in Lu and Dessouky [2004b]. This will reduce the complexity of the mathematical problem. By taking advantage of this relaxation the problem complexity now equals the one of a multi vehicle routing problem without pickup and delivery according to Lu and Dessouky [2004b].

An explicit sub-tour elimination is not necessary since it is implicitly included in the constraints (5) - (7).

The minimum workload or customer constraint As proposed, the original model was extended to incorporate the workload or customer spread considerations. For this purpose an additional set of decision variables and constraints is required.

The applied concept could also be used for other resource or capacity restrictions. For each resource a decision variable and constraints of size $2n+m+1$ is required. The following variables and the constants are required for the implementation of the constraints:

q_i ... the used amount of some limited resource at node i

ε_i ... the resources consumed by i - this could also be c_{ij}

$\overline{R_i}$ ... maximum workload or resource per route from depot i

$\underline{R_i}$ ... minimum workload or resource per route from depot i

The constraint sets consists of:

$$q_j \leq q_i + \varepsilon_i - M(1 - x_{ij}) \qquad \forall i \in N \backslash \{2n + m + 1\}, j \in N \tag{17}$$

$$q_j \geq q_i + \varepsilon_i + M(1 - x_{ij}) \qquad \forall i \in N \backslash \{2n + m + 1\}, j \in N \tag{18}$$

$$q_i - q_{i-1} \leq \overline{R_i} \qquad \forall i \in W \backslash \{2n + 1\} \tag{19}$$

$$q_i - q_{i-1} \geq \underline{R_i} \qquad \forall i \in W \backslash \{2n + 1\} \tag{20}$$

$$q_i \geq 0 \qquad \forall i \in N \tag{21}$$

The constraint set (17) and (18) are required to derive the current amount of the regulated resource along the route. Those constraints were both required in a min or max scenario. Constraint (19) is required if a resource is restricted (max), while 20 is required if some kind of workload distribution is applied (min). This formulation could also be applied if the resource constant is a decision variable too.

The following chapters assume a workload allocation constraint where a certain number of jobs per vehicle is defined. The required resource constant ε_i equals 1 if the node is a pickup node and the constant $\underline{R_i}$ equals the number of customers or jobs required per route or depot.

4.2 Set partitioning formulation

One of the methods applied is a column generation algorithm. As other algorithms applied to similar problems, the path based formulation applied in the master problem is a set covering formulation based on a set of valid paths. While the flow formulation focuses on the routing decision, the set partitioning formulation focuses on selecting the cost optimal set of routes from a given set of valid routes.

For the set partitioning formulation the following new constants, decision variables, and sets were required:

$c_{\gamma,d}$ ... the costs of path y when serviced by depot d

Γ ... the set of valid paths - defined by the sub problem

$\alpha_{i,\gamma}$ ... a constant that indicates if customer i is on route γ

$y_{\gamma,d}$ ... decision variable indicating if path γ is in the solution and which depot d is used to service the route

Corresponding to the flow formulation, the objective it to minimize the routing costs. In the path formulation this is considered with:

$$\min \sum_{\gamma \in \Gamma, d \in D} y_{\gamma,d} c_{\gamma,d} \tag{22}$$

In addition the following constraints have to be considered:

$$\sum_{\gamma \in \Gamma, d \in D} y_{\gamma,d} \alpha_{i,\gamma} = 1 \qquad \forall i \in N \backslash W \tag{23}$$

$$\sum_{\gamma \in \Gamma} y_{\gamma,d} = 1 \qquad \forall d \in W \tag{24}$$

$$y_{\gamma,d} = \{0,1\} \qquad \forall \gamma \in \Gamma, d \in W \tag{25}$$

The first constraint ensures that every customer or job is serviced once. The other constraint makes sure that every depot is used once. Both constraints can be relaxed as well as the decision variable. By relaxing constraint (23) to greater or equal and constraint (24) to smaller or equal, the problem complexity can be decreased. In addition, the decision variable $y_{\gamma,d}$ can be linearised since the problem structure will ensure integer solutions at all times.

5 Methods

The following chapter presents the methods applied to the problem at hand. Those will be benchmarked against each other and a MIP solver (IBM Cplex) implementation in the last Chapter 6. Starting from a Mixed Integer Problem (MIP), based on the better performing TSP formulation, multiple exact approaches were tested.

The problem at hand shares the most similarities with classical TSPs. So methods performing well for those as Branch and Cut in multiple compositions based on the work of Lu and Dessouky [2004b] and Laporte [1992] were tested first. The Branch and Cut algorithms applied included methods like lazy constraints as well as different types of callbacks used in the implementation of the proposed user cuts. The valid equalities and inequalities applied are adaptations of those presented in the literature and they are able to outperform predefined Branch and Cut algorithm of the MIP solver (IBM CPLEX) easily.

But the self proclaimed target to solve all proposed instances in less than 30 min could not be met. The second set of methods tested and proposed are two Benders Decomposition based algorithms. For the proposed problem a "classical" Benders Decomposition and a logic-based Benders Decomposition were implemented. The more common "classical" and the logic-based approach are described in Chapter 5.2.2. The row based decomposition seems to be the most efficient approach for the classical problem setting at hand.

After testing multiple compositions of the problem, especially with the workload constraint, the weaknesses of the approaches used became obvious.

The approaches that performed well on the less restrictive instances were not able to efficiently solve the more restrictive problem variations. Therefore, workload constraint seem to have a strong impact on the efficiency of the different approaches. In real world problems the solution space will be more restrictive than in those investigated here. One reason could be the presence of time window restrictions or similar features. For those reasons and for the purpose of completeness also decomposition by the columns was tested. The applied approach can be seen as worst case scenario since additional restrictions for the routing would be highly beneficial for the algorithm used.

The following sub chapters are dedicated to the description and configuration of the methods applied and tested. Computational results for those methods under different parameter setting are discussed in the Chapter 6 computational results.

5.1 Branch and Cut

With the intention of generating an efficient Branch and Cut approach for the problem at hand, multiple combinations of classical components were tested. All of the pieces used were previously proposed in the literature and adapted to the specifics of the problem presented above.

The following components were tested: lazy constraints for sub-tour elimination and for certain parts of the original vehicle flow formulation as well as multiple user defined cuts. All of the applied cuts or very similar versions were priviously described in Lu and Dessouky [2004a] and Ropke and Cordeau [2009]. Minor changes and adaptations are necessary to adapt some of the cuts to the problem at hand. A major part of the cuts presented in the literature for similar problems is not accessible for the problem at hand. The cuts based on capacity or time window restrictions could not be used but they seem to be a major key to the performance of the most solution approaches in the literature. In addition to the cuts and lazy constraints the impact of sub-tour elimination via lazy constraints was tested.

In the next chapter the basic problem setting will be discussed, the sub-tour elimination used as well as cuts and lazy constraints applied were discussed. At the end of this chapter the combinations tested for the computational experiments are defined. Before discussing the actual method, the following paragraphs will briefly discuss two general concepts: how a branch and bound procedure works and the difference between a lazy constraint and a user cut. Those concepts are important for understanding the actual methods applied.

Branch and Bound In general the Branch and Bound procedure is an algorithm or part of an algorithm that is used to systematically enumerate the solution space. The method was originally proposed by Land and Doig [1960] and introduced to routing problems by Little et al. [1963]. The proposed concepts were also used in other methods as for example Branch & Cut or Branch & Price. The basic concept applied in a Branch & Bound procedure is to 'divide and conquer'. The algorithm iteratively solves and creates subsets of the original solution space.

The Branching procedure describes the separation procedure that splits the current solution space into multiple new subsets. Each of those new subsets would be represented by a branch originating from the current solution. Those branches (usually) describe disjoint subsets of the previously solved subset. The number of branches created from a current solution depends on the branching strategy and the problem type. Applying the branching procedure recursively on its own would mean that the algorithm explores the whole solution space. So in addition another strategy is required to create an efficient strategy.

The Bounding procedure describes a way to reduce the number of branches (disjoint subsets) that are required to evaluate and further on branched. Two values are necessary to identify the nodes that can be eliminated, those are called bounds. No matter if one deals with a minimisation or a maximisation problem one may have to deal with a current lower and a current upper bound. But the definition of those varies depending on the problem type.

For a maximisation problem the (tightest or best) lower bound is the best so far identified solution. This solution has to be a feasible solution to the

overall problem, but is probably not the optimal solution. The upper bound is the objective value of a subset that is evaluated based on the information so far accessible. The objective value of the problem subset is an upper bound to the original problem since the the degree of freedom for it is greater (or equal) than the one of the original problem. The upper bound may be not a feasible solution to the original maximisation problem, but the subsets created based on the current one will never create a solution with a higher solution value.

On the other side for a minimisation problem the upper bound is the (tightest or best) so far identified feasible solution to the original problem. The lower bound is here the objective value of a subset of the original problem that may be infeasible concerning the original problem. But the lower bound is the best possible outcome of all subsets that may be created based on the subset evaluated.

The bounds were then used to identify those subsets that should be observed and used further on. Some subsets can be eliminated based on the knowledge provided by the bounds.

For a minimization problem those subsets that currently have an higher objective value than the current upper bound can be eliminated and do not have to be explored further on. On the other hand, for maximisation problems, subsets that are lower than the current lower bound may be eliminated. For both problem types the upper and the lower bound restrict the solution space that contains the optimal solution of the original problem. The upper bound of maximisation problems and the lower bound of minimisation problems is based on a mathematical relaxation of the original problem or the

solution of an heuristic. The lower bound of a maximisation problem and the upper bound of a minimisation problem is always a solution to a feasible and the so far best solution concerning the original problem.

Each subsets solved in the branch and bound is a mathematical relaxation or generated by an heuristic. Since the problem at hand applies an exact algorithm, the problem relaxation is applied here. There are multiple strategies on how to relax the original problem depending on its type. The kind of relaxation applied will also influence the bounding procedure applied as well as how many new branches may be or have to created from each of the current subsets.

In general a certain part of the original problem definition is ignore in the beginning. Probably the most common strategy is to reduce the problem complexity by relaxing the definition of the decision variables. One option would be to consider the decision variables to be linear instead of integer or binary.

Since the problem proposed previously is a binary problem (routing decision based on vehicle flow formulation) the applied procedure will be discussed briefly as an example for one of the branching schemes that will be applied. The solutions of the relaxed problem may contain non-binary decision variables. If one of the decision variables is not binary, branching is required. Two branches will be created. One represents a certain decision variable to be one, the other sets it to zero. Which one to pick for the branching depends on the strategy applied. The branching will define new subsets based on the previous one that only differentiates from the previous one by the definition of the decision variable used for the branching. All subsets

further on created in that branch will take the decisions made in previous branches into account. The problems get more restricted since the number of decision variables previously defined increases with every level of the branch and bound tree reached. Branching has to be applied until a feasible solution with respect to the original problem is found. The feasible solution found is a upper bound when solving a minimization problem and a lower bound when solving a maximisation problem. The current one is the one most restrictive to the solution space.

The optimal solution is found when all branches are solved or eliminated based on the bounding procedure. The gap which is the difference between an upper and a lower bound is then zero. The quality of the solution created within the branch and bound algorithm can be judged based on the gap.

Which branches to evaluate and eventually branch next is defined by a branching strategy. Those strategies differ in their focus. Some may find feasible solutions other good bounds earlier. This algorithm or principals can be combined with other strategies as further on described.

Lazy Constraint vs. (User) Cuts For the approach presented in this chapter two general concepts were used: user defined cuts and lazy constraints. Both of these constraints were added only if required. For every node in the Branch and Bound tree, the lazy constraints and the user cuts were checked and added, if the solution is not fulfilling one of these constraints. But in addition to those similarities they have a major difference. Lazy constraints are part of the problem definition while the user defined cuts are only used to strengthen the existing problem definition.

Algorithm 1 Pseudo code for a Branch and Cut algorithm applied on an integer minimization problem

```
bestObj=NULL;
branchSet=∅;
while branchSet not empty do
  select a branch from branchSet (depending on branching strategy);
  remove selected branch from branchSet;
  generate a subset problem branch for selected decision variable being 0
  and one being 1;
  solve the relaxed subset problem to obtain obj;
  if obj is bigger than upperbound then
    continue
  else if solution is binary then
    bestObj is obj
  else
    add branches to branchSet
  end if
end while
```

Based on those traits, lazy constraints are constraints from the original problem formulation. But instead of adding all of them in the beginning of the solution process, one will only add them to the problem definition when required. Theoretically you could use any constraint as lazy constraint that is part of the original problem definition. The evaluation process for the lazy constraints is rather time intensive and will be repeated until all violated constraints are identified. Therefore, an assessment between keeping constraints in the initial constraint set or introducing them to the set of lazy constraints is rather important. Those constraints that are binding in a high number of potential solutions should most probably be defined in the initial constraint set. Those constraints are only relevant in a few of the potential solutions, thus it is more beneficial to define them as lazy constraints. A

classical example in routing for a lazy constraint is the sub-tour elimination constraint.

User defined cuts on the other side are not an element in the original problem formulation. They are used to strengthen the problem formulation originally defined. By adding them, a certain proportion of the solution space can be discarded. The cuts try to eliminate parts of the linear solution that are invalid for the integer solution space. By doing so, the number of nodes required to generate the optimal integer solution within the branch and bound tree may be decreased.

5.1.1 General Concept of a Branch and Cut

The branch and cut approach is very similar to the branch and bound approach. The main difference is the way the solution in the different nodes of the branch and bound tree get evaluated.

Every time a candidate solution is found, the algorithm checks if it is valid concerning the lazy constraints. If a lazy constraints is not fulfilled, its added to the problem definition and the problem associated to the node in the branch and bound tree gets re-evaluated. When its not required to add additional lazy constraints a valid solution to the linear problem is found.

In a next phase we test the found solution for the proposed cuts. The general principal is similar to the one applied with the lazy constraints. Whenever a constraint defined as one of the cuts is not ensured, it gets added to the linear problem and the problem gets re-evaluated until a solution is obtained that fulfils all cuts proposed.

When a valid solution for the current node concerning the lazy constraints and the cuts is identified, that is also integer, the final node of the branch is found. It is not required to create new offspring of that branches. If the solution is linear the branching procedure gets applied.

In the branch and cut approach presented here, the vehicle flow formulation presented earlier was used. The initial model is only reduced by those constraints defined as lazy constraints and the binary condition for the decision variables get relaxed to generate a linear problem.

For the problem presented here the sub tour elimination is done before the lazy constraints or the precedence constraints were tested. Since applying sub tour elimination may reduce the number of precedence constraints required. The lazy constraints and cuts applied are presented further on.

The conditions for an optimal solution are the same as in a branch and bound procedure. The reason for this is the link of the integer and the linear problem formulation. The linear problem formulation will always be the lower bound to the integer solution. The best integer solution found within the branch and bound tree is the valid upper bound. In general when the upper and lower bound get the same solution and therefore their gap is zero, optimality is proven. So when no node with a linear solution is left that is below the best inter solution that represents the current upper bound, the best integer solution is proven optimal since the linear solution can not be improve any further.

5.1.2 Cuts

The following cuts were applied to the linear relaxations of the original problem in every node of the Branch and Bound algorithm:

Transfer Constraints: The proposed constraint ensures that pickup nodes get visited before the associated delivery node can be routed. Node i has to be visited before node $n + i$.

$$b_{n+i,h_1} + \sum_{j}^{k-1} b_{h_j,h_{j+1}} + b_{h_k}, i \leq k \tag{26}$$
$$\forall i \in (h1, \ldots, h_k) \in N \backslash \{i, 2n + m + 1\}, 1 \leq k \leq |N| - 2$$

Adjacent restrictions: This constraints strengthen the precedence constraints by checking the requirement for pairs of directly connected nodes. Whenever a pickup node i is visited before some node k, the delivery node "i+n" has to be visited after k.

$$b_{k,i} + b_{k,i+n} \geq x_{i,k} + x_{k,i} \qquad \forall i \in P, k \in N \backslash \{i, i + n\} \tag{27}$$

$$b_{i,k} + b_{i+n,k} \geq x_{i+n,k} + x_{k,i+n} \qquad \forall i \in P, k \in N \backslash \{i, i + n\} \tag{28}$$

Pairing restrictions: Also taking in consideration the pickup and delivery restrictions, the following constraints ensures for every job pair that they were scheduled for the same depot.

$$\sum_{k \in N} b_{ki} + 1 \geq \sum_{k \in N} b_{k,i+n} \qquad \forall i \in P \tag{29}$$

Demand restrictions: In most PDPs there is a certain demand associated to each of the jobs. The pickup locations are associated with the same demand since they are paired. For the purpose of deriving the capacity for every node, it is required to associate the pickup node with a positive demand and the equivalent delivery node with a negative demand of the same value.

Making use of this characteristics Lu and Dessouky [2004a] present a cut that can be applied to strengthen the problem formulation at hand even if it does not incorporate demands. Lu and Dessouky [2004a] try to take advantage of the demand definition in the mathematical model to make sure the pickup and the delivery node were associated with the same depot (in our case it would be same depot).

The idea is that only a very few combinations will be able to result in the same equilibrium than the one resulting from having the pickup and delivery nodes of the paired jobs on the same vehicle. This cut depends on a high level of deprivation in demand between the different nodes. If the nodes were fairly similar, the cuts have only a small impact to the solution procedure since combinations of non paired nodes could also fulfil the constraint.

Since here such a set of demands is not given, artificial demands were associated to each of the jobs. The artificial demand has to be unique for this purpose and will benefit from being not constructable from a set of other demands. The easiest way to find such a set of demands is using prime numbers. Since the number of jobs is limited, the number of known primal numbers is more than enough to ensure the uniqueness of every possible pairing in this constraint. This trick might even be beneficial for those PDPs

with demands since the constraint gets more restrictive by doing so without eliminating any valid solution.

The following constant is required for the the proposed cut:

d_k ... originally the demand of node k; for pickup positive and negative for delivery

The constant would be defined a primal number in our definition or setting for the cut. The cut 30 is just taking the sum over all demands served on the same vehicle or by the depot.

$$\sum_{k \in N} (b_{ki} d_k) = 0 \qquad \forall i \in W \tag{30}$$

Proof: All cuts can be proven by contradiction. For the proof one may consult Lu and Dessouky [2004a] or one of the other papers presenting those or similar constraints.

5.1.3 Lazy Constraints

The concept behind a lazy constraint is to introduce certain constraints only when required. In a lot of mathematical problems there are certain constraints that exist in high numbers even if they are not binding in the most of the solution steps. In some cases it might be better to add those constraints only when required. By adding those constraints only when required the solution time can be decreased. For the implementation at hand two kinds of lazy constraints were used.

Sub Tour Elimination The concept of sub tour elimination is an elementary problem in the context of routing. A sub tour is when the solution contains cycles instead of offering a continuous route for every vehicle. The sub tour elimination comes for free in problems like those using time window restrictions or similar concepts. In the problem at hand the precedence constraints offer free sub tour elimination.

The general model is ensuring that the solutions are free from sub tours and cycles. But since the constraints responsible for the sub tour elimination are used as lazy constraints, it might be beneficial to test for sub tours with lazy constraints to speed up the solution process.

The constraint added to the problem when required is:

$$\sum_{e \in E(S)} x_e \leq |S| - 1 \tag{31}$$

The lazy constraint applied will enforce for every sub set of nodes S that the sum of connections inside this subset is smaller than the size of the sub set.

Precedence Constraints Based on the assumption, that those constraints that are present in high numbers but only binding in a small portion of solution steps and in accordance to Lu and Dessouky [2004a], the precedence constraints were used as lazy constraints.

Lu and Dessouky [2004a] used constraints (6) to (7) as lazy constraints, but after pretests it seemed to be more beneficial to only transfer the con-

straints (6) and (7) from the set of normal constraints to the set of Lazy constraints.

5.1.4 Tested Compositions

The last chapter presented multiple methods commonly used in Branch and Cut. Three different versions of the Branch and Cut were tested to identify which parts of the algorithm used are most promising:

- **Cuts and Lazy Constraints and Sub tour Elimination:** This method includes the full range of features implemented. Every time a solution is found it is first tested for the sub tour elimination constraints and afterwards for the precedence constraints. If a sub tour elimination constraint is added the precedence constraints were not tested to save computational efforts. When both groups of lazy constraints were checked and the found solution was a valid solution the cuts were also tested and added if required. The constraints (6) and (7) are not present in the initial model.

- **Cuts and Lazy Constraints:** Since the precedence constraints will also eliminate sub tours, the sub tour elimination was excluded from the process, due to the fact that the high number of possible constraints was to expensive in terms of computational time. Since the problem at hand is not requiring explicit sub tour elimination, the set of initial constraints stays the same.

- **Cuts:** The most basic version tested is the one using the full vehicle flow formulation in combination with the proposed cuts.

For all of the methods presented here an additional trick was used to improve the run time. As Lu and Dessouky [2004a] proposed and also in accordance with modern standards in solver, those constraints dominated by other constraints were purged to reduce the constraint set. By doing so the number of constraints can be decreased with a low amount of additional computational effort. The methods proposed by the used software (IBM CPLEX) were used without any adaptations.

In Figure 3 the run times for the different methods were plotted. On average the best results could be obtained, using only the proposed cuts and starting with the original complete problem formulation.

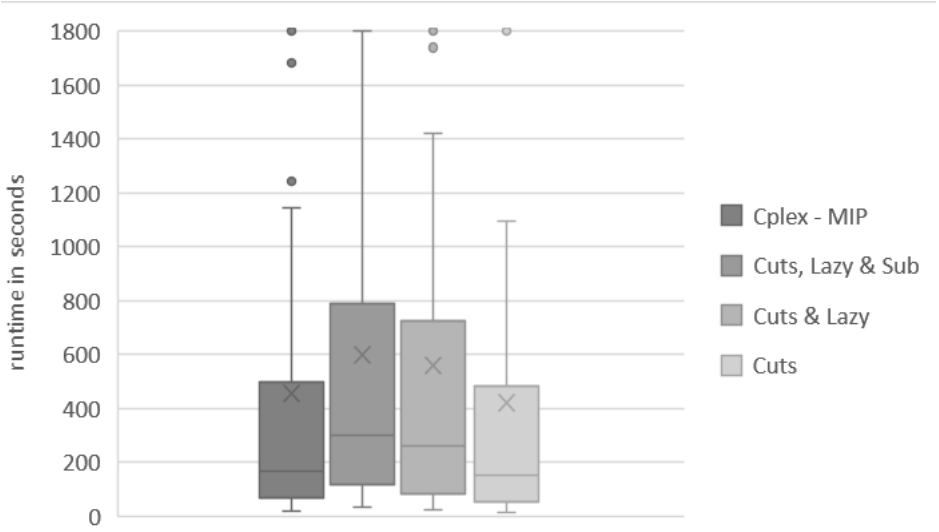


Figure 3: Box-plot comparing the run times in seconds for different Branch-and-Cut variants

5.2 Benders Decomposition

The following section will discuss an alternative decomposition approach within a branch and bound framework. For the first method proposed, the Branch and Cut explained in the previous chapter, additional constraints (so called cuts) were used to strengthen the linear relaxations in every node of the branch and bound procedure. In this chapter Benders Decomposition is used to strengthen the linear relaxation. In every node the linear relaxations of the original problem including the fixed integer values from the branch and bound procedure were solved with a Benders Decomposition approach. Benders Decomposition used to iteratively explore and restrict the solution space.

The so called Benders Decomposition was originally proposed by J.F. Benders a Dutch mathematician in Benders [1962]. Embedding Benders Decomposition in a Branch and Bound/Cut approach is for example described in Sridhar and Park [2000].

In the following sections the general concepts applied in the Benders Decomposition approaches, the master problem, and the sub problems tested are described. For the problem at hand one master problem was tested with two different sub problems. The main difference between the two sub problems tested are the cuts used in combination with the proposed sub-problem: Logical Benders Cuts and "classical" Benders Feasibility Cuts.

5.2.1 General Concept

The general concept used in Benders Decomposition is to solve multiple smaller problems at a time instead of one big problem at once. The original problem gets divided in multiple stages with each of those containing a certain set of constraints and decision variables. Most of the Benders Decomposition approaches consist of only two stages but in general there is no limit to the number of stages one may apply. But it is necessary to assign all decision variables of the same type to the corresponding stages.

For the general explanations only two stages are assumed to be used. This is the classical setting of a Benders Decomposition based algorithm as well as the setting used for the actual implementation. The two stages will be iteratively solved. The first stage contains the so called master problem while the other stages contain so called sub-problems. When ever a solution of the first stage is obtained, the second stage is solved based on the first stage solution in order to validate the first stage solution. As long as the first stage solution results is an invalid second stage solution new constraints will be added to the associated first (or next higher) stage problem. Those iteratively added constraints are based on the second stage information and called Benders cuts. The algorithm tries to find an optimal solution to the original problem by considering only as many constraints as required.

Since this algorithm adds additional constraints until an optimal solutions is obtained, it is also called row generation (the contrary concept of column generation that will be discussed in the next chapter). In a matrix representation of a mathematical model, every column will represent a variable

and every row a constraint. An entry in the matrix represents the constant associated to a certain decision variable. If the constant is zero the decision variable associated to that column is not present in the constraint. Adding a constraint will add an additional row to the matrix representation. The number of rows and therefore the size of the matrix will increase with every iteration until the optimal solution is found.

The optimal solution is found when the first stage solution leads to a valid second stage solution and then it is not necessary to add additional Benders Cuts. The effectiveness of a Bender Decomposition approach is highly dependent on the structure of the problem. Choosing the right set of constraints and decision variables for each of the stages is crucial for an effective Benders Decomposition approach.

5.2.2 Formulation

The problem described in Chapter 4.1 has two (main) decision variables, one for the routing decision and one for the precedence relations. Since the precedence relations will restrict the routing decisions, they are an obvious choice for the second stage problem.

For the problem at hand the first stage will generate routes. Those routes do not incorporate the precedence constraints and may contain sub tours. The second stage will ensure that those solutions containing sub-tours and not scheduled pickup nodes before the paired delivery nodes, will be iteratively withdrawn from the solution space.

By dividing the problem in a master problem with a routing decision and a sub problem with a scheduling decision, similarities to the concept of

Sexton and Bodin [1985a] and Sexton and Bodin [1985b] can be seen. Sexton and Bodin [1985a] solved a single vehicle routing problem that incorporated desired delivery times.

Minimum Customer Constraint: Considering the minimum customer constraint in the solution method is fairly simple. The required constraints and decision variables can either be add to the first (master problem) or second stage problem. A small set of tests suggested that using the first stage (master problem) might be the better option. Since the complexity and the number of constraints as well as decision variables is limited, the difference for the first stage (master problem) solution process is rather small. By doing so the second stage model as well as the cuts were not affected by the change.

Master Problem: The master problem is a routing problem of TSP kind without any form of sub tour elimination. The original problem formulation of the first stage will only ensure that every customer is entered once and left once as defined in the following formulation:

$$min \sum_{(i,j)\in A} x_{ij} c_{ij} \tag{32}$$

$$s.t. \sum_{(i)\in N} x_{ij} = 1 \qquad \forall j \in N \tag{33}$$

$$\sum_{(j)\in N} x_{ij} = 1 \qquad \forall i \in N \tag{34}$$

$$x_{ij} = \{0, 1\} \qquad \forall (i, j) \in A \tag{35}$$

The solutions generated like this are then tested within the second stage problem.

Second Stage: For the second stage, one model was used in combination with two different cuts. The second stage of the Benders decomposition approach applied is based on the constraint (5) - (7). The solutions that are infeasible in terms of the precedence constraints should be identified and then excluded by adding additional Benders Cuts. The first stage problem will be iteratively resolved with a growing set of constraints until a feasible solution is identified. When the second stage problem proves the feasibility of an first stage solution the optimal solution is identified.

When applying Benders Decomposition the second stage is a dual problem based on the original formulation ((5) - (7)). The second stage formulation as primal, based on the first stage solution $x\prime$, is defined as:

$$min\ 0 \tag{36}$$

$$s.t\ b_{ki} \leq b_{kj} + (1 - x_{ij}\prime) \tag{37}$$

$$\forall (i,j) \in A/\{2n+m+1, 2n+1\}, k \in N\backslash\{i\}$$

$$b_{kj} \leq b_{ki} + (1 - x_{ij}\prime) \tag{38}$$

$$\forall (i,j) \in A/\{2n+m+1, 2n+1\}, k \in N\backslash\{i\}$$

$$x_{ij}\prime \leq b_{ij} \qquad \forall (i,j) \in A \tag{39}$$

A disadvantage to consider is that a few of the constraints, that were previously ensured implicitly, have to be considered explicitly when decom-

position is applied. Those constraints are (7) - (16) from the original problem formulation.

Benders Cuts: The Benders Cuts applied are actually feasibility cuts that exclude invalid solutions from the routing problem. First stage solutions may be invalid because they are not taking the precedence constraints into account or include sub tours.

Two types of those feasibility cuts were tested: logic-based Benders Cuts and "classical" Benders Feasibility Cuts. Logic based Benders Decomposition is introduced and described for binary problems in Hooker and Ottosson. The "classical" Benders Feasibility Cuts using the dual information of the second stage problem are well explained in Sridhar and Park [2000].

Both cuts try to exclude solutions similar to the ones that previously resulted in invalid second stage problems. The "classical" Benders Feasibility cuts will exclude a whole area of the first stage solution space by introducing cuts based on the dual information of the second stage problem. Since the second stage problem is a dual problem, its dual information can be used to restrict the first stage solution space.

The logic-based Benders Decomposition approach will only exclude the previously chosen solution instead of the whole area. To exclude the invalid solutions from the first stage problem the logic-based Benders Decomposition approach introduces the following cuts:

$$\sum_{(i,j)\in A: x_{ij}\prime=1} x_{ij} + \sum_{(i,j)\in A: x_{ij}\prime=0} (1 - x_{ij}) \tag{40}$$

The "classical" Benders Decomposition approach seems to be far superior to the logic-based approach in terms of solution time. The simplicity that is offered with the logic-based approach comes with an high wastage in terms of obtained information. Both options will solve the same basic problem but the "classical" Benders Feasibility Cuts will most probably exclude bigger portions of the first stage solution space at a time. This makes "classical" Benders Feasibility Cuts more competitive and the method of choice for the further investigations.

5.3 Column Generation

The concept behind Column Generation again could be described as a divide and conquer strategy. Instead of solving one big flow problem and reducing runtime with a wide range of sophisticated tricks, it might be more beneficial to solve two easier problems at a time. In a Column Generation approach a predefined number of constraints (rows) is given and fixed but the number of decision variables (columns) will increase over time. This is the exact opposite to the Benders Decomposition approach shown in the previous chapter.

Column Generation separates the routing decision (path creation) from the path selection. The path selection is done by the so called master problem. In the problem presented here the master problem is a linear version of the path based formulation proposed in Chapter 4.2. The routing decision is a shortest path problem with additional restrictions. Important for such an approach is how to define or identify a promising route which is beneficial to generate and consider in the set partitioning problem. The common approach to do this are the dual values associated to the decision if a certain node or customer is considered to be introduced to a certain route.

The required components for a column generation approach will be discussed in the following sub chapter: the general algorithm, the master problem (route selection) used, the method to solve the sub problem (path generation problem) as well as methods to improve the performance such as dominance rules.

5.3.1 General description

As previously proposed the column generation approach requires two parts: a method to identify and select promising paths and a way to generate new once. Both of this parts interact with each other. Over multiple iterations they try to detect new paths that might decrease the objective value of the master problem until no further improvement can be achieved.

The algorithm starts with an initial set of feasible solutions. Those can be generated with a limited run of the proposed solution method for the sub problem or with any other method (e.g. an heuristic). The routes generated with the methods of choice were introduced to the master problem which is a linear relaxation of the original optimization problem. By solving the so called master problem one obtains dual information for each of the constraints that help to identify new candidate solutions for the master problem. For the identification of new routes there are multiple options as well: heuristics, mathematical programming (shortest path problem), or labelling algorithms.

In general all methods suitable to solve the shortest path problem can be applied to generate new paths in the sub problem. As long as the last iterations of the sub problem were solved with an exact approach, the found solution is proven optimal. When solving the master problem the costs used are based on the negative reduced costs from the master problem. The dual values indicate how much it might save to introduce a certain node into a route. The sub problem to solve is not a shortest path problem based on the initial distance, but rather a shortest path problem based on the actual routing costs as well as the dual values or negative reduced costs obtained

in the master problem in the last iteration. While generating new routes or solving the sub problem, the new potential solutions get evaluated with the sum of their routing costs minus the sum of the dual values of every node on the route. When adding a new set of paths to the master problem those were added to the master problems objective function with the actual routing costs. The actual routing costs are those that are not depending on the dual information. In general, all routes with negative reduced costs have to be identified since those have the potential to decrease the current objective value of the master problem.

When all or a sufficiently sized set of new paths is generated, the set is added to the master problem and the master problem is solved once again. The process of solving the master problem and generating new paths will be repeated as long as new paths can be identified. When an exact approach such as labelling or an MIP is not able to identify new paths, the optimal solution is considered to be found. It is possible to use an exact or an heuristic approach to generate new routes within the column generation. But optimality can only be proven when at least one full run of an exact approach is applied to the sub problem that is not able to find new paths with negative reduced costs.

In addition to those basic parts some useful addition can be included to decrease column generations run time: starting with a beneficial set of paths in the first iteration, applying dominance rules that decrease the number of paths in the sub problem, and the correct assessment of the amount of time spend at each iteration of the sub problem.

5.3.2 Master Problem

The master problem used for the Column Generation approach is the previously presented path based formulation of the MDTSPPD. The objective is to minimize the cots while servicing all of the nodes or customers. Each of the depots is allowed or forced to service one route. For the original problem formulation without workload considerations empty routes are present in the initial set. Every depot has to service one route, only if a workload (or a similar) constraint is present. The master problem has two purposes: selecting a beneficial set of routes based on the set of generated routes and offering insides which routes might be a good complement for the existing or previously identified set of routes (dual information).

5.3.3 Sub Problem and Labelling

The sub problem at hand is a variation of a classical shortest path problem. Additional requirements arise from the paired pickup and delivery jobs, the multiple depots, and the workload requirements contained in the extensions.

The basis concept used for the labelling algorithm is similar to other column generation approaches. Starting from the origin (a depot) one gradually expands the route by a new node or customer. Every time doing so one has to check the feasibility of the newly created path. Upper capacity limits can be checked after each extension while lower limits can only be checked at the end depot. The distances and the dual costs of the label get updated. Every time a route gets extended the new and the existing routes will be compared through dominance rules and those paths dominated will get discarded.

When a path gets extended back to the depot the dual costs associated with the depot will be added as well. All paths that can be extended to the final depot and that are not dominated throughout the labelling algorithm as well as fulfil all minimum constraints (e.g. min workload) will be added to the master problem further on.

The dominance criteria reduces the required amount of paths to be considered within the labelling algorithm. Most of the time that is spend in the labelling algorithm will be spend with checking those labels. Reducing the number of active paths will directly reduce the time spend in the associated routines. The labelling algorithm is the most time consuming element in the whole column generation approach. Improving its efficiency will drastically improve the run time performance of the overall algorithm.

An easy to implement strategy for the multi depot problem is to deal with the paths of all (three) depots at once. By introducing an artificial depot with a distance of zero to each of the customers the basic shortest path problem with pickup and delivery can be used. But every path that gets extended until the final depot has to be extended to each of the depots, then checked for dominance, and afterwards added to the master problem if not dominated.

5.3.4 Dominance Rules and their Relaxation

Since the exact labelling approach will find multiple permutations of equivalent or similar routes, it is beneficial to invest the available solution time only on those with the highest potential (heuristic runs) or those required to be checked (exact runs).

Contardo and Martinelli [2014] present a set of dominance rules considering multiple depots and Cordeau et al. [2008] present a set of rules that can be applied to pickup and delivery problems. By combining those two sets of dominance rules, a set applicable to the MDTSPPD can be obtained.

The set of dominance rules applied when solving the sub problem of the MDTSPPD is defined as:

$$x.costs \leq \quad y.costs \tag{41}$$

$$x.nodesVisited \supseteq \quad y.nodesVisited \tag{42}$$

$$x.openRequests \subseteq \quad y.openRequests \tag{43}$$

$$x.lastNodeOfTour = \quad y.lastNodeOfTour \tag{44}$$

$$x.depotUsed = \quad y.depotUsed \tag{45}$$

A route x may dominate another route y in a pairwise comparison when its not more expensive than the other, visited at least the same set of nodes, having the same set of nodes still to visit (pickup nodes visited define a delivery node to include), and using the same depot (or dummy) for it. The number of routes can be reduced by just dealing with the routes without any depot connection. This reduces the number of routes to test by the factor of up to three. For the implementation two dummy depots were used that represent the start and the end of a tour. The actual depots were only considered when the routes were extended to the dummy end depot while being not dominated. Dominated tours get not considered further on. The pairwise comparison is done whenever a tour is extended to a new node. All

previously created non dominated nodes were checked that end on the same node.

Most of the algorithm's time is spent at the dominance rules. So, they have to be applied efficiently. Route x can be dominated by route y if one (or more) criteria are not met. Every dominance rule will require one comparison per route tested. By sorting them according to the probability that they are not fulfilled, one may save running time. Those with the lowest probability to be fulfilled were checked first. Identifying the appropriate sequence is done with a small set of test cases. Be reminded that by using bit vectors the complexity of comparing the costs of a route is equivalent to the comparison of two sets. The structure of the tests ensures that the last node of the tour is the same.

The dominance rules were tested in the following order:

If the extension of the route was to the end depot it gets checked first. Otherwise the costs were the first criterion followed by the sets for the nodes visited and the open requests. The nodes visited include pickup and delivery nodes. The size of the vector is neither limited nor impacts the solution time.

As proposed by Contardo and Martinelli [2014] the dominance rules can be applied heuristically. By applying only a subset of presented dominance criteria the computational efforts in early stages of the Column Generation approach may be reduced. For the proposed problem no improvement could be obtained by reducing the set of dominance rules used in early stages of the labelling algorithm. Hence, for the tested Column Generation approach all dominance rules proposed were present at all times.

6 Computational Results

The following chapter will compare the proposed methods in terms of their average runtime for the original problem formulation and the extended problem formulation with workload constraints. In addition, the costs of introducing the minimum workload constraints in terms of the average runtime and the increase in routing costs will be discussed.

The comparison will be performed based on the instances proposed by Berger and Bierwirth [2010]. Those instances will be described in the next section.

6.1 Instances

The test instances used in the following chapters were designed by Berger and Bierwirth [2010] for the Collaborative Carrier Routing Problem. They are generated based on the Solomon instance R101. The Solomon instances were originally generated for the Vehicle Routing Problem with time windows. Berger and Bierwirth [2010] took instance R101 which contains 101 nodes. Three of the Nodes were defined as depots: 10, 54 and 93. In the original Problem definition by Berger and Bierwirth [2010] node pairs were assigned to one of the depot consisting of a pick and delivers location. Each instance consists of nine jobs (18 nodes). Three jobs were associated with each of the depots. For the MDTSPPD it is not important to which depot a job was originally assigned. But discussing the optimal solutions later on will require some knowledge of the process during which they were created in the first place. For each job associated to one of the depots Berger and Bierwirth

[2010] selected two nodes from a certain region of the node space. The node space is defined by two things: the depot and instance class. Three different classes were created: Set A, Set O and Set I. Set A is strongly clustered. Nodes were picked from areas that are only associated to one of the depots. Set O is slightly more clustered. There is a certain area that is accessible for jobs of two (or even three) depots. This should enhance the collaborative potential. The last Set I is picking the pickup and delivery location randomly from all nodes of the R101 Solomon instance.

The node distribution of the R101 Solomon instance as well as the partition can be seen in the Figure 4.

Solution Characteristics: The Collaborative Carrier Routing Problem (or the MDTSPPD) is a problem without capacity or time restriction. Its solution characteristics were mainly driven by: the existences of multiple depots, the level of clustering in the instance class, and the paired pickup and delivery requests. As Figure 5 shows, for most of the instances created by Berger and Bierwirth [2010] solutions with only one route are optimal.

This behaviour results from certain characteristics of the problem at hand. For problems of this kind with only one depot it is easy to access that solutions with only one route being optimal in the original setting. Since the tour length is not limited by any kind of capacity constraint a Hamiltonian tour will be the optimal solution. The pickup and delivery request are no reason to use multiple tours.

But the effect of the multiple depots has to be discussed. When only multiple depots would be present without the pickup and delivery feature,

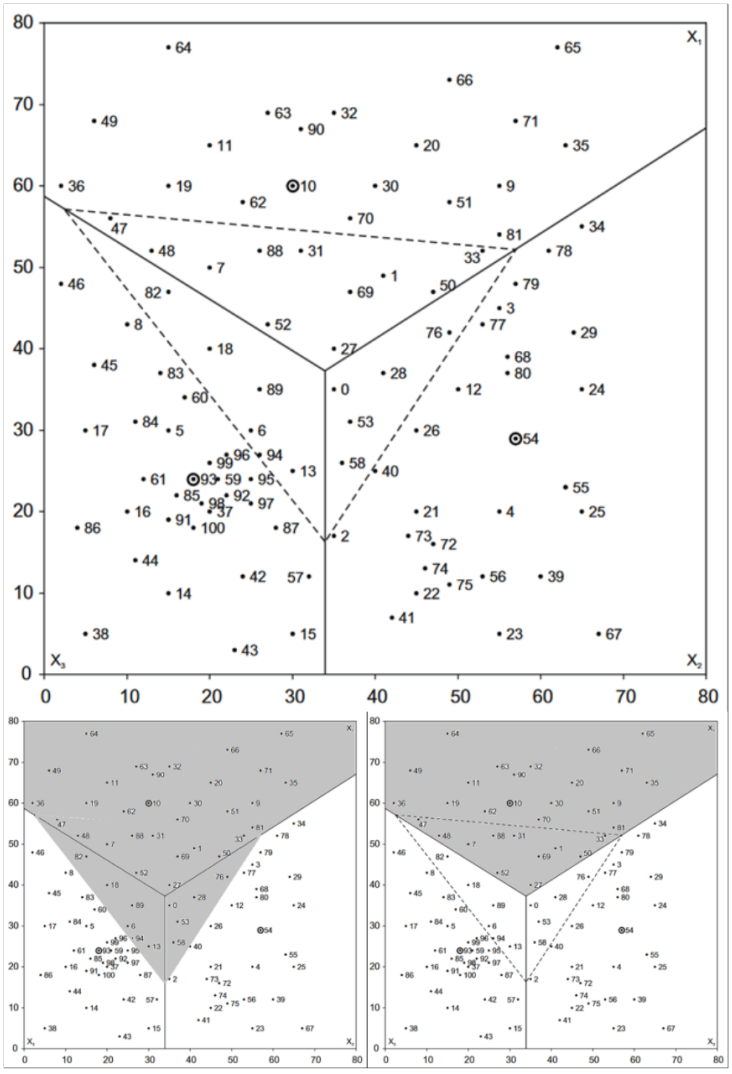


Figure 4: Instance schema for the instances used

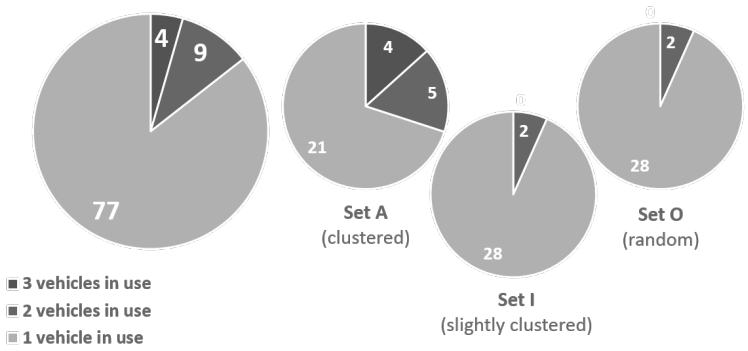


Figure 5: Vehicles used in each of the instances

the closeness of the nodes and depots will be elemental to determine the probability of solutions with routes from more than one depot. For more clustered instances the chances will be higher than for those with less clustered neighbourhoods.

By considering pickup and delivery in a multi depot scenario, the chances for multiple routes will decrease. The main reason for that is that pickup nodes that are in one region might force routes to also access other regions of the instance and therefore overcome the clustering.

Therefore the introduction of the workload constraint will have a negative impact on the solution quality in the most cases as explained later on.

6.2 Efficiently solving the MDTSPPD

In this part of the computational results section the performance of the proposed solution methods on the original instances by Berger and Bierwirth [2010] (without workload constraints) will be discussed. We will start off by identifying the most promising and robust variant of the previously proposed Branch-and-Cut algorithms. In a second step also the other exact

approaches, namely Benders Decomposition (with Feasibility Cuts) and Column Generation, were considered.

Branch and Cut As proposed by Lu and Dessouky [2004a] the Branch-and-Cut approach included Lazy constraints as well as explicit sub tour elimination. Although those mechanisms seemed to be beneficial for the problem formulation proposed in Lu and Dessouky [2004b], for the problem at hand this seems to be different.

For the problem at hand there is no approach of the three proposed once that is able to dominate the other two for all the instances proposed by Berger and Bierwirth [2010]. From the proposed 90 instances the approach including the lazy constraints as well as explicit sub tour elimination was the fastest for seven of the instances, the approach using only the lazy constraints was the best in twenty instances, and the approach only using the cuts was dominating the other to approaches in 59 instances. In terms of average run times the version only using cuts is far superior to the other versions. The good performance in terms of average run times per instance set can be observed in Table 1 as well as in the boxplot in figure 3. The results show that the distribution or clustering of the instances has an impact on the performance of the solution methods. With less clustering the application of lazy constraints and sub tour elimination in the Branch-and-Cut seem to become more competitive. But even in the completely randomly distributed instances the Breach-and-Cut without explicit sub tour elimination and lazy constraints will outperform the other two versions in the most of the instances. In addition, using only the cuts seems to have the least disad-

vantages in terms of run times compared to the fastest one in the particular instances if it is not the fastest method. On the other side, in almost all cases, where the Branch-and-Cut could outperform the other to method, it does it by a fairly high percentage.

In addition to the differences in run times, there were also minor differences concerning the number of instances that could not be solved by each of the approaches. Overall just three instances could not be solved within the time limit of 30 minutes by at least one of the proposed approaches. The approach applying lazy constraints, explicit sub tour elimination, and cuts could solve two instances that could not be solved by the other two approaches. Using cuts and lazy constraints could only solve one instance that the other two could not solve. And the far superior method also concerning the number of instances solved, was the version only applying the cuts that strengthen the formulation. This approach was able to solve four additional instances compared to the other to versions of the Branch-and-Cut.

Overall the Branch-and-Cut focusing on the cuts seems to be the most promising constellation of the three proposed Branch-and-Cut approaches in terms of average run times as well as number of instances that could be solved. The reason for such a behaviour is that (at least on the instances used for the tests) both types of lazy constraints are often binding constraints. Since they are the only constraint restricting the problem this behaviour is not surprising. For further comparisons the focus will be put on the best performing Branch-and-Cut approach. The one used for further analysis is the one only applying (user) cuts to strengthen the original formulation.

Table 1: Average runtime in seconds of CPLEX and the three proposed Branch-and-Cut variants; MIP = CPLEX, LC = Lazy Constraints, SE = Subtour Elimination, BC = Branch and Cut, and oC = only Cuts.

		Branch and Cut		
Average for	MIP	LC & SE	LC	oC
Set A (clustered)	257.04	341.93	332.73	195.94
Set I (slightly clustered)	434.36	599.67	599.14	329.00
Set O (random)	677.87	850.27	747.19	703.99
For all sets	456.42	597.29	559.69	397.68

Solving the MDTSPPD The most promising approaches or versions from the set of Branch-and-Cut, Benders Decomposition, and Column Generation variations were identified. Now those will be tested on the original Berger and Bierwirth [2010] instances.

In Table 2 the average performance in terms of run times of CPLEX (as reference), the best performing Branch-an-Cut, the Benders Decomposition, and the Column Generation approach applied are shown. The Benders Decomposition approach outperforms the other methods based on the average run time for all three instance types. Also in terms of instances that could not be solved within the time limit of 30 minutes, the Benders Decomposition approach is performing well with only three instances that could not be solved. Those three also could not be solved with one of the other solution approaches.

Important to note is that the Column Generation approach was not able to solve any of the instances within 30 minutes. The fastest three instances with Column Generation required 2928, 4000, and 6747 seconds. Those three instances could be solved by Benders decomposition with less than 2 seconds

each. For the rarely restricted solution space of the original problem formulation, the Column generation approach is clearly unable to provide competitive results. But this is not surprising, since Column Generation based methods are more beneficial for solution spaces with a small set of valid solutions. The solution space here is the exact opposite and the amount of solutions that are invalid is only restricted by the precedence constraints. The proposed changes, to add workload constraint, will restrict the solution space and therefore be beneficial for Column Generation based methods.

Table 2: Average runtime in seconds of CPLEX (MIP), Branch-and-Cut (BC), Benders Decomposition (BD), and Column Generation (CG) for each instance set.

Average for	MIP	BC	BD	CG
Set A (clustered)	257.04	195.94	27.35	-
Set I (slightly clustered)	434.36	321.00	179.42	-
Set O (random)	677.86	703.99	434.02	-
For all sets	456.42	397.68	213.60	-

6.3 Minimum workload constraints

This part of the computational results is dedicated to the changes that result from introducing the minimum workload constraints to the original problem formulation by Berger and Bierwirth [2010]. Two aspects will be analysed: the solution time and general performance of the proposed methods under different workload settings as well as the change in terms of costs that arise from those constraints.

Solution time The average solution time increases with increasing minimum workloads for the most of the methods. But for one methods the more restricted solution space is beneficial, Column Generation is able to become more competitive for the more restrictive scenarios. For the unrestricted and only slightly restricted scenario, the Benders Decomposition approach proposed seems to be the method of choice. But while the other approaches at least double their average solution time when increasing the workload restriction from zero to one third, column generation decreases its by more than a third. The most restrictive scenarios asked for at least two third of the amount of customers originally assigned to each of the carrier. In that setting the Benders decomposition approach gets even slower and therefore is the approach that takes the longest to solve the proposed instances. The Branch-and-Cut approach is able to maintain its average solution time compared to the previous scenario. The new best performing method is Column Generation that once more was able to reduce its average solution time, by more than the factor ten. The average solution times for the proposed methods under different workload settings can be found in Table 3.

Table 3: Average runtime in seconds of CPLEX (MIP), Branch-and-Cut (BC), Benders Decomposition (BD), and Column Generation (CG) for all instances and three different settings of the minimum workload constraint.

Min Customers	MIP	BC	BD	CG
no min customer	456.04	456.42	213.60	-
1/3 of the customer	1286.44	1023.72	432.72	-
2/3 of the customer	1689.62	1023.01	1056.88	566.55

The number of instances that the proposed approaches can solve within the time limit behave similarly. CPLEX is be able to solve only ten of the ninety instances in the more restrictive cases. Benders Decomposition and Branch-an-Cut solve slightly less than half of the instances. The interesting part is the behaviour when the minimum workload restriction is further increased. CPLEX and Benders Decomposition were able to solve less instances the more restrictive the minimum workload constraint got. The Branch-and-Cut approach loses a lot of its power when workload constraints were first introduced to the model. But, the number of instances that could be solved seems to be relatively robust against further changes. The Column Generation approach is the most competitive one in the most restrictive setting. It can solve all instances in the most restrictive setting. So those approaches behave similar in terms of average run times and number of instances that could be solved. The detailed changes in the number of instances that could be solved can be found in Table 4.

Table 4: Number of instances that could not be solved with each of the methods; MIP = CPLEX, BC = Branch and Cut, BD = Benders Decomposition, and CG = Column Generation.

Min Customers	MIP	BC	BD	CG
no min customer	11	11	4	90
1/3 of the customers	51	40	12	90
2/3 of the customers	80	41	43	0

Worth to notice is also the impact of the clustering in the different instance sets. Their impact of the clustering for the most restrictive scenario can be seen in Table 5. As previously in the other settings the less clustered

and therefore more typical scenarios for a courier service are much harder for the most methods than the clustered once. Only Column Generation is able to to solve all three instance types in approximately the same average run time. If there is a difference for Column Generation based on the level of clustering in the instance then Column Generation seems to be able to solve the apparently harder instances (based on the performance of the other methods) even faster than the apparently easier once.

Table 5: Average runtime in seconds of CPLEX (MIP), Branch-and-Cut (BC), Benders Decomposition (BD), and Column Generation (CG) for each instance set with a 2/3 min workload constraint.

(2/3 min Customer)

Average for	MIP	BC	BD	CG
Set A (clustered)	1494.48	382.74	484.67	596.05
Set I (slightly clustered)	1793.03	1256.32	1099.20	593.51
Set O (random)	1781.33	1429.95	1586.76	510.09
For all sets	1689.62	1023.01	1056.88	566.55

Costs of workload distribution The other interesting part when analysing the results from introducing this additional set of constraints is: How much does it cost to share the workload between the different carriers?

The results for the original problem setting without workload constraint have shown that in most of the solutions only one or sometimes two vehicles will be used. Based on this observation it is obvious that the costs will increase.

The introduction of a minimum of one third of the customers will already result in an cost increase of more than 18%. If the value even increases to

two thirds the cost will increase on average by 30%. Another interesting result is that for those instances that represent urban scenarios with higher competition, the cost increase is even higher with almost 25% respectively over 45% on average for the two workload scenarios. Additional information can be found in Table 6.

Table 6: Increase in total cost resulting from workload constraints

Average for	no min customer	1 / 3 of cust.	2 / 3 of cust.
Set A (clustered)	270.49	12.64%	21.87%
Set I (slightly clustered)	263.00	17.41%	28.49%
Set O (random)	267.49	24.91%	45.52%
For all sets	267.00	18.32%	31.96%

One may conclude two important things from those numerical experiments. The method that should be applied depends on two things: the level of clustering and the presence of workload constraints (or similar concepts). For the most part Benders decomposition will be the method of choice. But for special cases, namely those with a high level of clustering or with very restrictive workload constraints, Column Generation might be the only feasible choice. But keep in mind that those special cases will be in real world application the more common scenario. The best scenarios in real world applications with a high probability of collaboration will be those markets with highly competitive situations and those with almost no clustering.

The other important information that can be extracted from the proposed information is the impact of the workload constraints. The amount of profit that could be ruined by insisting on minimum workloads is pretty high and will increase with the level of competition between the different carriers (less

clustered). The most harm will be done by workload constraints in the scenarios that are most promising for collaboration, those that are less clustered and therefore more competitive.

7 Conclusion

The thesis at hand investigated a routing problem from the sphere of collaborative carrier routing with a centralized decision maker. In the collaborative carrier routing problem request, that consist of a pickup and a delivery location, were reassigned between the different carrier to decrease the costs and therefore increase the profit for all participants. Those problems are often solved decentralized because competitors hesitate to share their information and therefore their competitive advantage. Here a centralized planning was assumed to investigate the maximal gain that collaboration could achieve.

In addition to the basic problem formulation an extended formulation was proposed to consider an important real world feature: workload balancing. In real world application, solutions with unevenly distributed workloads are often considered unattractive for the participating carriers even when they are optimal in terms of costs.

Three different well-established exact solution methods and some variants for them were proposed. They were benchmarked without workload constraints and under different settings of the workload constraints.

The computational results show that for the benchmark instances in the most settings Benders Decomposition should be applied. But for more restricted scenarios with the proposed workload restrictions, the proposed Column Generation approach was able to dominate the other methods in all instances.

The results also show the strong impact of the workload constraints on the routing costs. Assigning each of the carriers, at least request equal to

one third of the amount on contracts, they originally added to the request pool, will increase the routing costs on average by 18%. For more competitive scenarios with less clustered structures, it will even increase up to 25%. Considering a workload of two thirds the cost increase will be more than 30% on average and even up to 45% for less clustered scenarios.

Based on the presented findings, future approaches for collaborative problems based on centralized or decentralized planning may be benchmarked. The insights generated on the performance of the different methods tested are useful when generating benchmarks for similar problems in science or real world application. By introducing and testing the proposed workload constraints in the given setting, it was shown how strong their impact on the potential gain were.

References

B. Adenso-Díaz, S. Lozano, S. Garcia-Carbajal, and K. Smith-Miles. Assessing partnership savings in horizontal cooperation by planning linked deliveries. *Transportation Research Part A: Policy and Practice*, 66:268–279, 2014. ISSN 0965-8564.

C. Archetti, M. G. Speranza, and D. Vigo. Vehicle routing problems with profits. *Vehicle Routing: Problems, Methods, and Applications*, 18:273, 2014. ISSN 1611973589.

R. Baldacci, E. Bartolini, A. Mingozzi, and R. Roberti. An exact solution framework for a broad class of vehicle routing problems. *Computational Management Science*, 7(3):229–268, 2010. ISSN 1619-6988. doi: 10.1007/s10287-009-0118-3. URL `http://dx.doi.org/10.1007/s10287-009-0118-3`.

R. Baldacci, E. Bartolini, and A. Mingozzi. An exact algorithm for the pickup and delivery problem with time windows. *Operations Research*, 59 (2):414–426, 2011.

J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, 4(1):238–252, 1962. ISSN 0029-599X.

G. Berbeglia, J.-F. Cordeau, I. Gribkovskaia, and G. Laporte. Static pickup and delivery problems: a classification scheme and survey. *TOP*, 15(1): 1–31, 2007. ISSN 1863-8279. doi: 10.1007/s11750-007-0009-0. URL `http://dx.doi.org/10.1007/s11750-007-0009-0`.

G. Berbeglia, J.-F. Cordeau, and G. Laporte. Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202(1):8–15, 2010. ISSN 03772217.

S. Berger and C. Bierwirth. Solutions to the request reassignment problem in collaborative carrier networks. *Transportation Research Part E: Logistics and Transportation Review*, 46(5):627–638, 2010. ISSN 1366-5545.

A. Bettinelli, A. Ceselli, and G. Righini. A branch-and-price algorithm for the multi-depot heterogeneous-fleet pickup and delivery problem with soft time windows. *Mathematical Programming Computation*, 6(2):171–197, 2014. ISSN 1867-2949.

P. Buijs, J. A. L. Alvarez, M. Veenstra, and K. J. Roodbergen. Improved collaborative transport planning at dutch logistics service provider fritom. *Interfaces*, 46(2):119–132, 2016. ISSN 0092-2102.

M. Cherkesly, G. Desaulniers, and G. Laporte. Branch-price-and-cut algorithms for the pickup and delivery problem with time windows and last-in-first-out loading. *Transportation Science*, 49(4):752–766, 2015. ISSN 0041-1655. doi: 10.1287/trsc.2014.0535.

C. Contardo and R. Martinelli. A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. *Discrete Optimization*, 12:129–146, 2014. ISSN 15725286. doi: 10.1016/j.disopt.2014.03.001.

J.-F. Cordeau and G. Laporte. The dial-a-ride problem: models and algorithms. *Annals of Operations Research*, 153(1):29–46, 2007. ISSN 1572-9338. doi: 10.1007/s10479-007-0170-8. URL `http://dx.doi.org/10.1007/s10479-007-0170-8`.

J.-F. Cordeau, G. Laporte, and S. Ropke. Recent models and algorithms for one-to-one pickup and delivery problems. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces*, pages 327–357. Springer US, Boston, MA, 2008. ISBN 978-0-387-77777-1. doi: 10.1007/978-0-387-77778-8{\textunderscore}15.

J.-F. Cordeau, M. Iori, G. Laporte, and J. J. Salazar González. A branch-and-cut algorithm for the pickup and delivery traveling salesman problem with lifo loading. *Networks*, 55(1):46–59, 2010. ISSN 00283045. doi: 10.1002/net.20312.

F. Cruijssen, O. Bräysy, W. Dullaert, H. Fleuren, and M. Salomon. Joint route planning under varying market conditions. *International Journal of Physical Distribution & Logistics Management*, 37(4):287–304, 2007. ISSN 0960-0035.

R. H. Currie and S. Salhi. Exact and heuristic methods for a full-load, multi-terminal, vehicle scheduling problem with backhauling and time windows. *Journal of the operational Research Society*, 54(4):390–400, 2003. ISSN 0160-5682.

B. Dai and H. Chen. Profit allocation mechanisms for carrier collaboration in pickup and delivery service. *Computers & Industrial Engineering*, 62 (2):633–643, 2012a. ISSN 03608352. doi: 10.1016/j.cie.2011.11.029.

B. Dai and H. Chen. Mathematical model and solution approach for carriers' collaborative transportation planning in less than truckload transportation. *International Journal of Advanced Operations Management*, 4(1-2): 62–84, 2012b. ISSN 1758-938X.

P. Detti, F. Papalini, and G. Z. M. de Lara. A multi-depot dial-a-ride problem with heterogeneous vehicles and compatibility constraints in healthcare. *Omega*, 70:1–14, 2017. ISSN 0305-0483.

R. Dondo and J. Cerdá. A cluster-based optimization approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows. *European Journal of Operational Research*, 176(3):1478–1507, 2007. ISSN 03772217.

A. G. Dragomir, D. Nicola, A. Soriano, and M. Gansterer. Multi-depot pickup and delivery problems in multiple regions: A typology. *International Transactions in Operational Research, forthcoming*, 2017.

Y. Dumas, J. Desrosiers, and F. Soumis. The pickup and delivery problem with time windows. *European Journal of Operational Research*, 54(1):7–22, 1991. ISSN 03772217. doi: 10.1016/0377-2217(91)90319-Q.

Ö. Ergun, G. Kuyzu, and M. Savelsbergh. Shipper collaboration. *Computers & Operations Research*, 34(6):1551–1560, 2007. ISSN 03050548.

M. Gansterer and R. F. Hartl. Request evaluation strategies for carriers in auction-based collaborations. *OR Spectrum*, 38(1):3–23, 2016. ISSN 1436-6304. doi: 10.1007/s00291-015-0411-1.

M. Gansterer and R. F. Hartl. Collaborative vehicle routing: a survey. *arXiv preprint arXiv:1706.05254*, 2017.

B. Golden, S. Raghavan, and E. Wasil, editors. *The Vehicle Routing Problem: Latest Advances and New Challenges*. Operations Research/Computer Science Interfaces. Springer US, Boston, MA, 2008. ISBN 978-0-387-77777-1. doi: 10.1007/978-0-387-77778-8.

J. N. Hooker and G. Ottosson. Logic-based benders decomposition. *Mathematical Programming*, 96(1):33–60. ISSN 1436-4646. doi: 10.1007/s10107-003-0375-9. URL `http://dx.doi.org/10.1007/s10107-003-0375-9`.

S. Irnich. A multi-depot pickup and delivery problem with a single hub and heterogeneous vehicles. *European Journal of Operational Research*, 122(2): 310–328, 2000. ISSN 03772217.

B. Kalantari, A. V. Hill, and S. R. Arora. An algorithm for the traveling salesman problem with pickup and delivery customers. *European Journal of Operational Research*, 22(3):377–386, 1985. ISSN 03772217. doi: 10.1016/0377-2217(85)90257-7.

M. A. Krajewska and H. Kopfer. Collaborating freight forwarding enterprises. *OR Spectrum*, 28(3):301–317, 2006. ISSN 1436-6304.

G. Kuyzu. Lane covering with partner bounds in collaborative truckload transportation procurement. *Computers & Operations Research*, 77:32–43, 2017. ISSN 03050548.

A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960. ISSN 00129682, 14680262. doi: 10.2307/1910129.

G. Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3): 345–358, 1992. ISSN 03772217. doi: 10.1016/0377-2217(92)90192-C.

C. K. Lin. A cooperative strategy for a vehicle routing problem with pickup and delivery time windows. *Computers & Industrial Engineering*, 55(4): 766–782, 2008. ISSN 03608352.

J. D. C. Little, K. G. Murty, D. W. Sweeney, and C. Karel. An algorithm for the traveling salesman problem. *Operations Research*, 11(6):972–989, 1963. doi: 10.1287/opre.11.6.972.

R. Liu, Z. Jiang, R. Y. K. Fung, F. Chen, and X. Liu. Two-phase heuristic algorithms for full truckloads multi-depot capacitated vehicle routing problem in carrier collaboration. *Computers & Operations Research*, 37 (5):950–959, 2010. ISSN 03050548.

Q. Lu and M. Dessouky. An exact algorithm for the multiple vehicle pickup and delivery problem. *Transportation Science*, 38(4):503–514, 2004a. ISSN 0041-1655. doi: 10.1287/trsc.1030.0040.

Q. Lu and M. Dessouky. An exact algorithm for the multiple vehicle pickup and delivery problem. *Transportation Science*, 38(4):503–514, 2004b. ISSN 0041-1655.

R. Masson, S. Ropke, F. Lehuédé, and O. Péton. A branch-and-cut-and-price approach for the pickup and delivery problem with shuttle routes. *European Journal of Operational Research*, 236(3):849–862, 2014. ISSN 03772217. doi: 10.1016/j.ejor.2013.08.042.

J. R. Montoya-Torres, A. Muñoz-Villamizar, and C. A. Vega-Mejía. On the impact of collaborative strategies for goods delivery in city logistics. *Production Planning & Control*, 27(6):443–455, 2016. ISSN 0953-7287.

S. Nadarajah and J. H. Bookbinder. Less-than-truckload carrier collaboration problem: modeling framework and solution approach. *Journal of Heuristics*, 19(6):917–942, 2013. ISSN 1381-1231.

G. Nagy and S. Salhi. Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. *European Journal of Operational Research*, 162(1):126–141, 2005. ISSN 03772217.

S. N. Parragh, K. F. Doerner, and R. F. Hartl. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58(1):21–51, 2008a. ISSN 1614-631X. doi: 10.1007/s11301-008-0033-7. URL `http://dx.doi.org/10.1007/s11301-008-0033-7`.

S. N. Parragh, K. F. Doerner, and R. F. Hartl. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58(2):81–117, 2008b. ISSN 1614-631X. doi: 10.1007/s11301-008-0036-4. URL `http://dx.doi.org/10.1007/s11301-008-0036-4`.

E. Pérez-Bernabeu, A. A. Juan, J. Faulin, and B. B. Barrios. Horizontal cooperation in road transportation: a case illustrating savings in distances and greenhouse gas emissions. *International Transactions in Operational Research*, 22(3):585–606, 2015. ISSN 1475-3995.

S. Ropke and J.-F. Cordeau. Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, 43(3):267–286, 2009. ISSN 0041-1655.

S. Ropke, J.-F. Cordeau, and G. Laporte. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks*, 49 (4):258–272, 2007. ISSN 00283045. doi: 10.1002/net.20177.

S. Salhi and G. Nagy. A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. *Journal of the operational Research Society*, pages 1034–1042, 1999. ISSN 0160-5682.

M. Sanchez, L. Pradenas, J.-C. Deschamps, and V. Parada. Reducing the carbon footprint in a vehicle routing problem by pooling resources from different companies. *NETNOMICS: Economic Research and Electronic Networking*, 17(1):29–45, 2016. ISSN 1385-9587.

M. W. P. Savelsbergh and M. Sol. The general pickup and delivery problem. *Transportation Science*, 29(1):17–29, 1995. ISSN 0041-1655.

T. R. Sexton and L. D. Bodin. Optimizing single vehicle many-to-many operations with desired delivery times: I. scheduling. *Transportation Science*, 19(4):378–410, 1985a. ISSN 0041-1655.

T. R. Sexton and L. D. Bodin. Optimizing single vehicle many-to-many operations with desired delivery times: Ii. routing. *Transportation Science*, 19(4):411–435, 1985b. ISSN 0041-1655.

V. Sridhar and J. S. Park. Benders-and-cut algorithm for fixed-charge capacitated network design problem. *European Journal of Operational Research*, 125(3):622–632, 2000. ISSN 03772217.

P. Toth, editor. *The vehicle routing problem: Problems, methods, and applications*. MOS-SIAM series on optimization. SIAM, Philadelphia, Pa., 2. ed. edition, 2014. ISBN 9781611973587. URL `http://dx.doi.org/10.1137/1.9781611973594`.

B. Turan, S. Minner, and R. F. Hartl. A vns approach to multi-location inventory redistribution with vehicle routing. *Computers & Operations Research*, 78:526–536, 2017. ISSN 03050548. doi: 10.1016/j.cor.2016.02.018.

L. Verdonck, A. N. Caris, K. Ramaekers, and G. K. Janssens. Collaborative logistics from the perspective of road transportation companies. *Transport Reviews*, 33(6):700–719, 2013. ISSN 0144-1647. doi: 10.1080/01441647.2013.853706.

X. Wang, H. Kopfer, and M. Gendreau. Operational transportation planning of freight forwarding companies in horizontal coalitions. *European Journal of Operational Research*, 237(3):1133–1141, 2014. ISSN 03772217.

L. Xue, Z. Luo, and A. Lim. Exact approaches for the pickup and delivery problem with loading cost. *Omega*, 59:131–145, 2016. ISSN 0305-0483. doi: 10.1016/j.omega.2015.05.012.

Abstract

The thesis is dealing with an optimization problem with multiple collaborating transportation service providers. In the recent past the conditions for the whole branch of transportation, especially for the delivery of packages, got increasingly demanding. The guaranteed delivery times got shorter and there is a constant price pressure. Those developments require new solution procedures. Collaboration and sharing the existing infrastructures may be one way to address those conditions.

Exact solution procedures are applied to a basic problem formulation from the literature and a new variant. The new variant introduces workload constraints. Those workload constraints make it possible to divide the overall workload between the different carriers, e.g. in order to increase the perceived fairness. For both variants exact solution methods have been developed, based on those presented in the literature for similar problems.

The objective of the thesis is to identify suitable solution methods for the different problem variants and to determine the costs arising from considering those workload constraints.

The experiments show that the method should be chosen based on the problem variant and the restrictiveness of the workload constraints and that the costs of introducing such constraints are considerably high.

Zusammenfassung

Die Masterarbeit beschäftigt sich mit einem Optimierungsproblem, welches bei der Zusammenarbeit mehrerer Tansportdienstleister auftritt. In den letzten Jahren sind die Anforderungen an die Logistikbranche, insbesondere jene im Bereich Pakettransport, immer weiter gestiegen. Die Verkürzung der zugesicherten Lieferzeiten und der stetige Preisdruck erfordern neue Lösungsansätze. Eine Zusammenarbeit durch gemeinsame Verwendung der vorhandenen Ressourcen über Unternehmensgrenzen hinaus ist eine Möglichkeit diesen Umweltumständen Rechnung zu tragen.

Es werden exakte Lösungsmethoden auf ein bereits in der Literatur vorgestelltes Modell als auch auf eine neue Variante dieses Problemes angewendet. Die neue Variante ermöglicht es die Arbeitslast zwischen den verschiedenen Teilnehmern aufzuteilen, um so zum Beispiel die wahrgenommene Fairnis zu erhöhen. Für beide Modellvarianten wurden verschiedene exakten Lösungsverfahren erarbeitet, basierend auf jenen Methoden, die in der Literatur für ähnliche Probleme vorgestellt wurden.

Ziel der Arbeit ist es für die verschiedene Problemvarianten geeignete Lösungsmethoden zu identifizieren und festzustellen, welche Kosten durch die Berücksichtigung von Nebenbedingungen zur Verteilung der Arbeitslast entstehen.

Die vorliegende Arbeit zeigt, dass die Wahl der Lösungsmethode von der Restriktivität der Nebenbedingung abhängig gemacht werden sollte und dass die Kosten, die durch die Berücksichtigung einer solchen Nebenbedingung entstehen, vergleichsweise hoch sind.